autogluon_each_location

October 29, 2023

1 Config

[1]:
```
# config

label = 'y'
metric = 'mean_absolute_error'
time_limit = 60*60
presets = "experimental_zeroshot_hpo_hybrid"#'best_quality'

do_drop_ds = True
# hour, dayofweek, dayofmonth, month, year
use_dt_attrs = []#["hour", "year"]
use_estimated_diff_attr = False
use_is_estimated_attr = True

drop_night_outliers = True
drop_null_outliers = False

# to_drop = ["snow_drift:idx", "snow_density:kgm3", "wind_speed_w_1000hPa:ms",␣
 ↪"dew_or_rime:idx", "prob_rime:p", "fresh_snow_12h:cm", "fresh_snow_24h:cm",␣
 ↪"wind_speed_u_10m:ms", "wind_speed_v_10m:ms", "snow_melt_10min:mm",␣
 ↪"rain_water:kgm2", "dew_point_2m:K", "precip_5min:mm", "absolute_humidity_2m:
 ↪gm3", "air_density_2m:kgm3"]#, "msl_pressure:hPa", "pressure_50m:hPa", ␣
 ↪"pressure_100m:hPa"]
to_drop = ["wind_speed_w_1000hPa:ms", "wind_speed_u_10m:ms", "wind_speed_v_10m:
 ↪ms"]

excluded_model_types = ['CAT', 'XGB', 'RF']

use_groups = False
n_groups = 8

# auto_stack = True
num_stack_levels = 0
num_bag_folds = None# 8
num_bag_sets = None#20
```

```
use_tune_data = True
use_test_data = True
#tune_and_test_length = 0.5 # 3 months from end
# holdout_frac = None
use_bag_holdout = True # Enable this if there is a large gap between score_val␣
 ↪and score_test in stack models.

sample_weight = None#'sample_weight' #None
weight_evaluation = False#
sample_weight_estimated = 1
sample_weight_may_july = 1

run_analysis = False

shift_predictions_by_average_of_negatives_then_clip = False
clip_predictions = True
shift_predictions = False
```

2 Loading and preprocessing

[2]:
```
import pandas as pd
import numpy as np



import warnings
warnings.filterwarnings("ignore")


def feature_engineering(X):
    # shift all columns with "1h" in them by 1 hour, so that for index 16:00,␣
 ↪we have the values from 17:00
    # but only for the columns with "1h" in the name
    #X_shifted = X.filter(regex="\dh").shift(-1, axis=1)
    #print(f"Number of columns with 1h in name: {X_shifted.columns}")


    columns = ['clear_sky_energy_1h:J', 'diffuse_rad_1h:J', 'direct_rad_1h:J',
               'fresh_snow_12h:cm', 'fresh_snow_1h:cm', 'fresh_snow_24h:cm',
               'fresh_snow_3h:cm', 'fresh_snow_6h:cm']

    # Filter rows where index.minute == 0
    X_shifted = X[X.index.minute == 0][columns].copy()

    # Create a set for constant-time lookup
    index_set = set(X.index)
```

```
    # Vectorized time shifting
    one_hour = pd.Timedelta('1 hour')
    shifted_indices = X_shifted.index + one_hour
    X_shifted.loc[shifted_indices.isin(index_set)] = X.
 ↪loc[shifted_indices[shifted_indices.isin(index_set)]][columns]

    # set last row to same as second last row
    X_shifted.iloc[-1] = X_shifted.iloc[-2]

    # Count
    count1 = len(shifted_indices[shifted_indices.isin(index_set)])
    count2 = len(X_shifted) - count1

    print("COUNT1", count1)
    print("COUNT2", count2)

    # Rename columns
    X_old_unshifted = X_shifted.copy()
    X_old_unshifted.columns = [f"{col}_not_shifted" for col in X_old_unshifted.
 ↪columns]

    date_calc = None
    # If 'date_calc' is present, handle it
    if 'date_calc' in X.columns:
        date_calc = X[X.index.minute == 0]['date_calc']




    # resample to hourly
    print("index: ", X.index[0])
    X = X.resample('H').mean()
    print("index AFTER: ", X.index[0])

    X[columns] = X_shifted[columns]
    #X[X_old_unshifted.columns] = X_old_unshifted

    if date_calc is not None:
        X['date_calc'] = date_calc

    return X



def fix_X(X, name):
```

```
    # Convert 'date_forecast' to datetime format and replace original column␣
 ↪with 'ds'
    X['ds'] = pd.to_datetime(X['date_forecast'])
    X.drop(columns=['date_forecast'], inplace=True, errors='ignore')
    X.sort_values(by='ds', inplace=True)
    X.set_index('ds', inplace=True)


    X = feature_engineering(X)

    return X



def handle_features(X_train_observed, X_train_estimated, X_test, y_train):
    X_train_observed = fix_X(X_train_observed, "X_train_observed")
    X_train_estimated = fix_X(X_train_estimated, "X_train_estimated")
    X_test = fix_X(X_test, "X_test")


    if weight_evaluation:
        # add sample weights, which are 1 for observed and 3 for estimated
        X_train_observed["sample_weight"] = 1
        X_train_estimated["sample_weight"] = sample_weight_estimated
        X_test["sample_weight"] = sample_weight_estimated


    y_train['ds'] = pd.to_datetime(y_train['time'])
    y_train.drop(columns=['time'], inplace=True)
    y_train.sort_values(by='ds', inplace=True)
    y_train.set_index('ds', inplace=True)

    return X_train_observed, X_train_estimated, X_test, y_train




def preprocess_data(X_train_observed, X_train_estimated, X_test, y_train,␣
 ↪location):
    # convert to datetime
    X_train_observed, X_train_estimated, X_test, y_train =␣
 ↪handle_features(X_train_observed, X_train_estimated, X_test, y_train)

    if use_estimated_diff_attr:
        X_train_observed["estimated_diff_hours"] = 0
        X_train_estimated["estimated_diff_hours"] = (X_train_estimated.index -␣
 ↪pd.to_datetime(X_train_estimated["date_calc"])).dt.total_seconds() / 3600
```

```
        X_test["estimated_diff_hours"] = (X_test.index - pd.
↪to_datetime(X_test["date_calc"])).dt.total_seconds() / 3600

        X_train_estimated["estimated_diff_hours"] =␣
↪X_train_estimated["estimated_diff_hours"].astype('int64')
        # the filled once will get dropped later anyways, when we drop y nans
        X_test["estimated_diff_hours"] = X_test["estimated_diff_hours"].
↪fillna(-50).astype('int64')

    if use_is_estimated_attr:
        X_train_observed["is_estimated"] = 0
        X_train_estimated["is_estimated"] = 1
        X_test["is_estimated"] = 1


    # drop date_calc
    X_train_estimated.drop(columns=['date_calc'], inplace=True)
    X_test.drop(columns=['date_calc'], inplace=True)


    y_train["y"] = y_train["pv_measurement"].astype('float64')
    y_train.drop(columns=['pv_measurement'], inplace=True)
    X_train = pd.concat([X_train_observed, X_train_estimated])


    # clip all y values to 0 if negative
    y_train["y"] = y_train["y"].clip(lower=0)

    X_train = pd.merge(X_train, y_train, how="inner", left_index=True,␣
↪right_index=True)

    # print number of nans in y
    print(f"Number of nans in y: {X_train['y'].isna().sum()}")

    print(f"Size of estimated after dropping nans:␣
↪{len(X_train[X_train['is_estimated']==1].dropna(subset=['y']))}")


    X_train["location"] = location
    X_test["location"] = location

    return X_train, X_test
# Define locations
locations = ['A', 'B', 'C']

X_trains = []
X_tests = []
```

```
# Loop through locations
for loc in locations:
    print(f"Processing location {loc}...")
    # Read target training data
    y_train = pd.read_parquet(f'{loc}/train_targets.parquet')

    # Read estimated training data and add location feature
    X_train_estimated = pd.read_parquet(f'{loc}/X_train_estimated.parquet')

    # Read observed training data and add location feature
    X_train_observed= pd.read_parquet(f'{loc}/X_train_observed.parquet')

    # Read estimated test data and add location feature
    X_test_estimated = pd.read_parquet(f'{loc}/X_test_estimated.parquet')

    # Preprocess data
    X_train, X_test = preprocess_data(X_train_observed, X_train_estimated,␣
 ↪X_test_estimated, y_train, loc)

    X_trains.append(X_train)
    X_tests.append(X_test)

# Concatenate all data and save to csv
X_train = pd.concat(X_trains)
X_test = pd.concat(X_tests)
```

```
Processing location A…
COUNT1 29667
COUNT2 1
index:  2019-06-02 22:00:00
index AFTER:  2019-06-02 22:00:00
COUNT1 4392
COUNT2 2
index:  2022-10-28 22:00:00
index AFTER:  2022-10-28 22:00:00
COUNT1 702
COUNT2 18
index:  2023-05-01 00:00:00
index AFTER:  2023-05-01 00:00:00
Number of nans in y: 0
Size of estimated after dropping nans: 4418
Processing location B…
COUNT1 29232
COUNT2 1
index:  2019-01-01 00:00:00
index AFTER:  2019-01-01 00:00:00
COUNT1 4392
COUNT2 2
```

```
index:  2022-10-28 22:00:00
index AFTER:  2022-10-28 22:00:00
COUNT1 702
COUNT2 18
index:  2023-05-01 00:00:00
index AFTER:  2023-05-01 00:00:00
Number of nans in y: 4
Size of estimated after dropping nans: 3625
Processing location C…
COUNT1 29206
COUNT2 1
index:  2019-01-01 00:00:00
index AFTER:  2019-01-01 00:00:00
COUNT1 4392
COUNT2 2
index:  2022-10-28 22:00:00
index AFTER:  2022-10-28 22:00:00
COUNT1 702
COUNT2 18
index:  2023-05-01 00:00:00
index AFTER:  2023-05-01 00:00:00
Number of nans in y: 6059
Size of estimated after dropping nans: 2954
```

2.1 Feature enginering

2.1.1 Remove anomalies

[3]:
```python
import numpy as np
import pandas as pd


# loop thorugh x train[y], keep track of streaks of same values and replace␣
 ↪them with nan if they are too long
# also replace nan with 0

import numpy as np

def replace_streaks_with_nan(df, max_streak_length, column="y"):
    for location in df["location"].unique():
        x = df[df["location"] == location][column].copy()

        last_val = None
        streak_length = 1
        streak_indices = []
        allowed = [0]
        found_streaks = {}
```

```
        for idx in x.index:
            value = x[idx]
            # if location == "B":
            #     continue

            if value == last_val and value not in allowed:
                streak_length += 1
                streak_indices.append(idx)
            else:
                streak_length = 1
                last_val = value
                streak_indices.clear()

            if streak_length > max_streak_length:
                found_streaks[value] = streak_length

                for streak_idx in streak_indices:
                    x[idx] = np.nan
                streak_indices.clear()  # clear after setting to NaN to avoid␣
 ↪setting multiple times
        df.loc[df["location"] == location, column] = x

        print(f"Found streaks for location {location}: {found_streaks}")

    return df


# deep copy of X_train into x_copy
X_train = replace_streaks_with_nan(X_train.copy(), 3, "y")
```

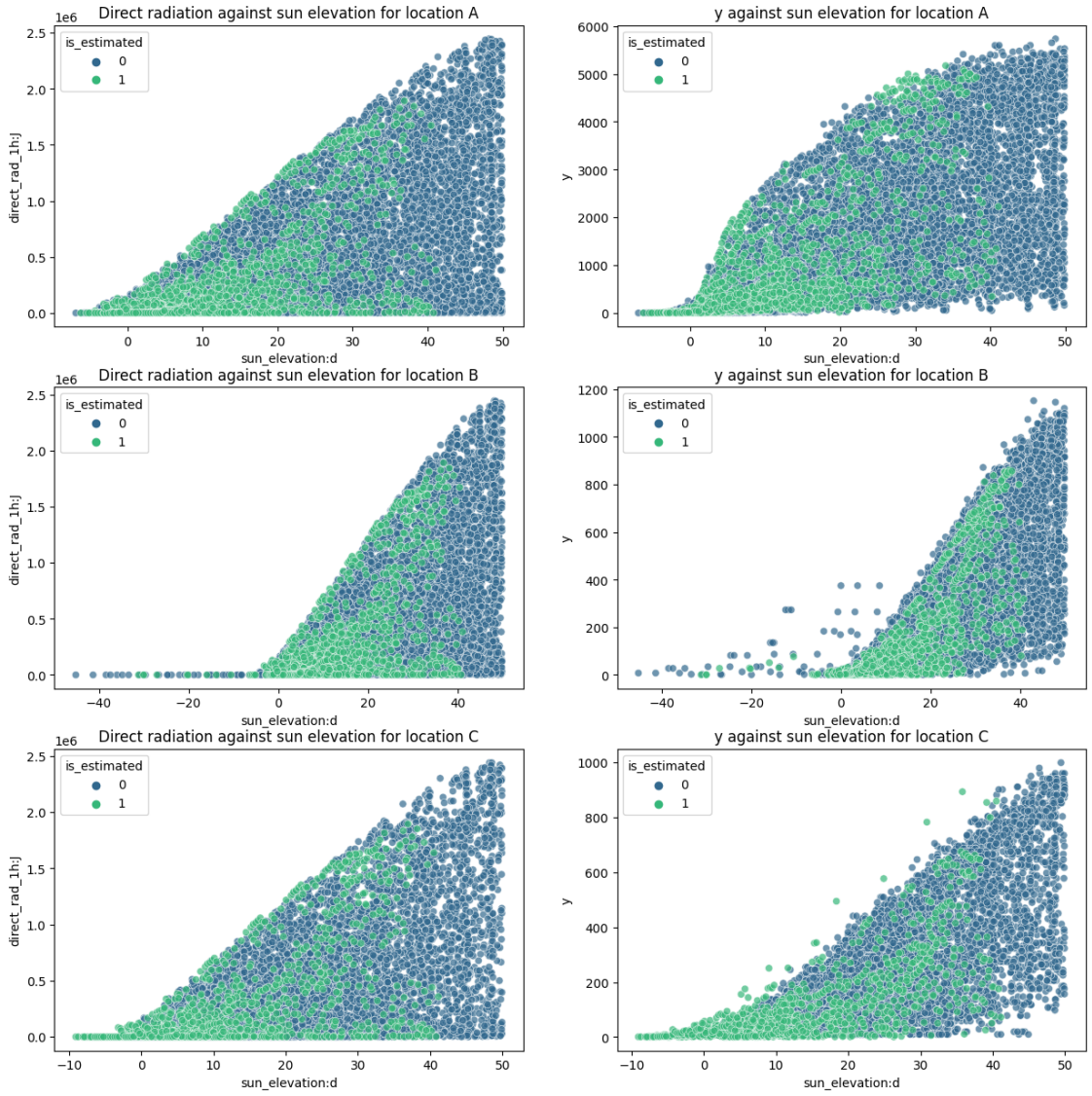
```
Found streaks for location A: {}
Found streaks for location B: {3.45: 28, 6.9: 7, 12.9375: 5, 13.8: 8, 276.0: 78,
18.975: 58, 0.8625: 4, 118.1625: 33, 34.5: 11, 183.7125: 1058, 87.1125: 7,
79.35: 34, 7.7625: 12, 27.6: 448, 273.41249999999997: 72, 264.78749999999997:
55, 169.05: 33, 375.1875: 56, 314.8125: 66, 76.7625: 10, 135.4125: 216, 81.9375:
202, 2.5875: 12, 81.075: 210}
Found streaks for location C: {9.8: 4, 29.400000000000002: 4, 19.6: 4}
```

[4]:
```
# print num rows
temprows = len(X_train)
X_train.dropna(subset=['y', 'direct_rad_1h:J', 'diffuse_rad_1h:J'],␣
 ↪inplace=True)
print("Dropped rows: ", temprows - len(X_train))
```

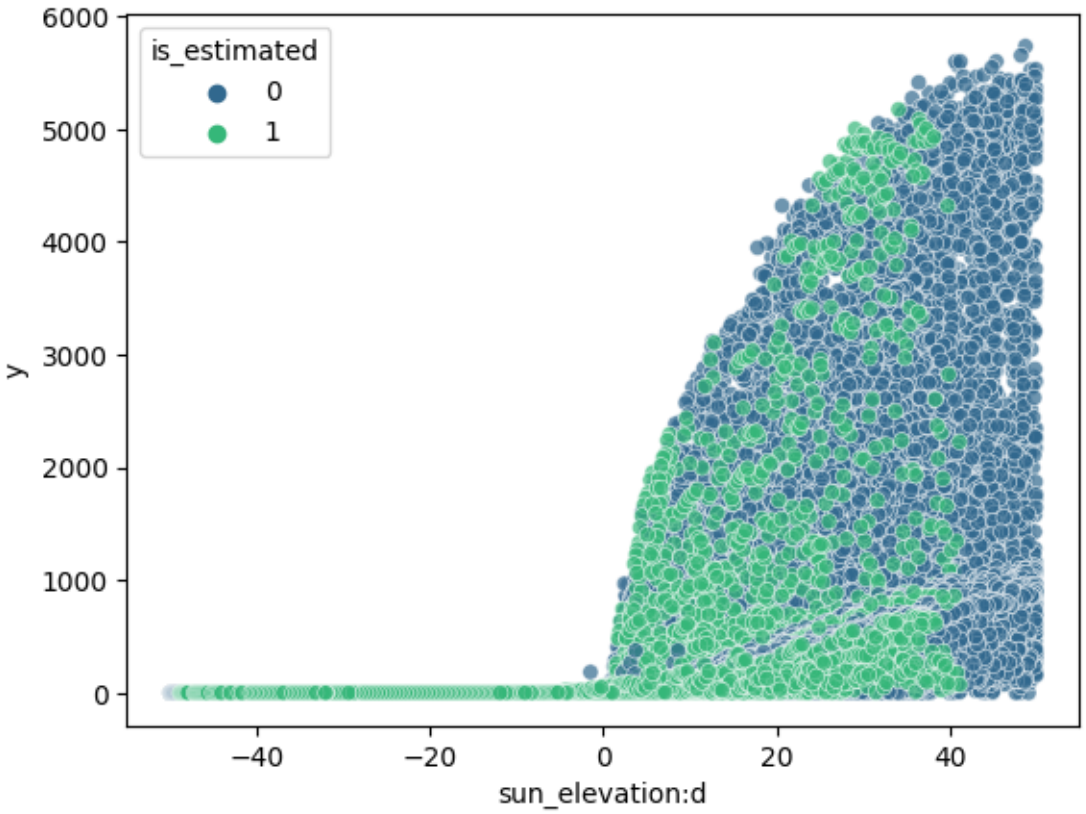
```
Dropped rows:  9293
```

[5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Filter out rows where y == 0
temp = X_train[X_train["y"] != 0]

# Plotting
fig, axes = plt.subplots(len(locations), 2, figsize=(15, 5 * len(locations)))

for idx, location in enumerate(locations):
    sns.scatterplot(ax=axes[idx][0], data=temp[temp["location"] == location],␣
 ↪x="sun_elevation:d", y="direct_rad_1h:J", hue="is_estimated",␣
 ↪palette="viridis", alpha=0.7)
    axes[idx][0].set_title(f"Direct radiation against sun elevation for␣
 ↪location {location}")

    sns.scatterplot(ax=axes[idx][1], data=temp[temp["location"] == location],␣
 ↪x="sun_elevation:d", y="y", hue="is_estimated", palette="viridis", alpha=0.7)
    axes[idx][1].set_title(f"y against sun elevation for location {location}")

# plt.tight_layout()
# plt.show()
```

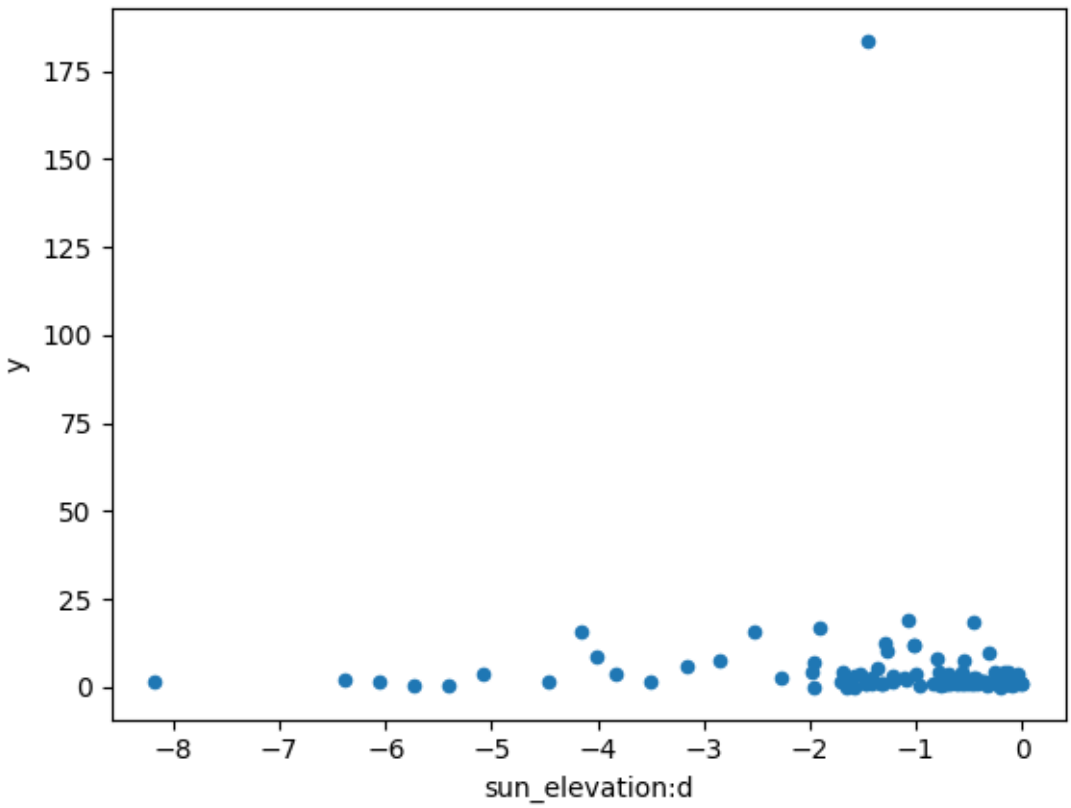


[6]:
```
thresh = 0.1

# Update "y" values to NaN if they don't meet the criteria
mask = (X_train["direct_rad_1h:J"] <= thresh) & (X_train["diffuse_rad_1h:J"] <=␣
 ↪thresh) & (X_train["y"] >= 0.1)
if drop_night_outliers:
    X_train.loc[mask, "y"] = np.nan

# Plot using sns scatterplot
sns.scatterplot(data=X_train, x="sun_elevation:d", y="y", hue="is_estimated",␣
 ↪palette="viridis", alpha=0.7)
plt.show()
```



[7]:
```
# location B count number of rows with y > 0 and sun_elevation:d < 0

condition = (X_train["location"] == "B") & (X_train["y"] > 0) &␣
 ↪(X_train["sun_elevation:d"] < 0)
bad = X_train[condition]

bad.plot.scatter(x="sun_elevation:d", y="y")
```

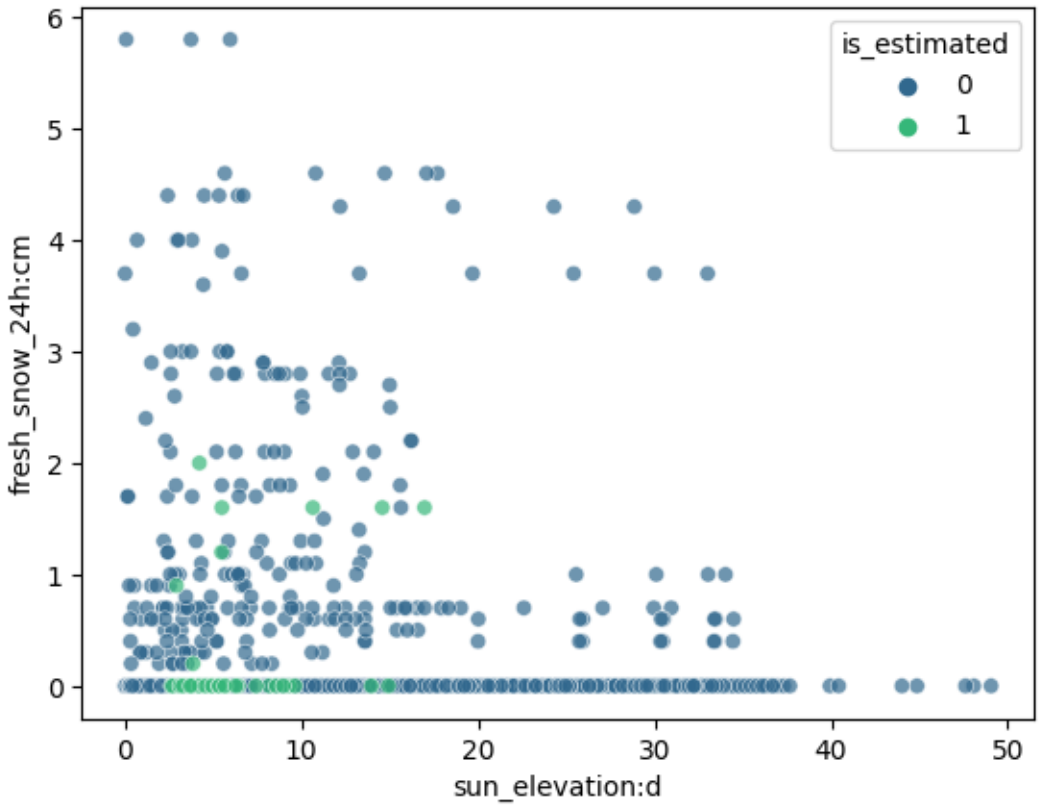
[7]: <AxesSubplot: xlabel='sun_elevation:d', ylabel='y'>



[8]:
```
# set y to nan where y is 0, but direct_rad_1h:J or diffuse_rad_1h:J are > 0␣
 ↪(or some threshold)
threshold_direct = X_train["direct_rad_1h:J"].max() * 0.01
threshold_diffuse = X_train["diffuse_rad_1h:J"].max() * 0.01
print(f"Threshold direct: {threshold_direct}")
print(f"Threshold diffuse: {threshold_diffuse}")

mask = (X_train["y"] == 0) & ((X_train["direct_rad_1h:J"] > threshold_direct) |␣
 ↪(X_train["diffuse_rad_1h:J"] > threshold_diffuse)) & (X_train["sun_elevation:
 ↪d"] > 0) & (X_train["fresh_snow_24h:cm"] < 6) & (X_train[['fresh_snow_12h:
 ↪cm', 'fresh_snow_1h:cm',  'fresh_snow_3h:cm', 'fresh_snow_6h:cm']].
 ↪sum(axis=1) == 0)
print(len(X_train[mask]))

#print(X_train[mask][[x for x in X_train.columns if "snow" in x]])

# show plot where mask is true
#sns.scatterplot(data=X_train[mask], x="sun_elevation:d", y="y",␣
 ↪hue="is_estimated", palette="viridis", alpha=0.7)
```

```
sns.scatterplot(data=X_train[mask], x="sun_elevation:d", y="fresh_snow_24h:cm",␣
 ↪hue="is_estimated", palette="viridis", alpha=0.7)
plt.show()

#sns.scatterplot(data=X_train[mask], x="fresh_snow_24h:cm",␣
 ↪y="total_cloud_cover:p", hue="is_estimated", palette="viridis", alpha=0.7)



# set y to nan where mask
if drop_null_outliers:
    X_train.loc[mask, "y"] = np.nan

# show how many rows for each location, and for estimated and not estimated
X_train[mask].groupby(["location", "is_estimated"]).count()["direct_rad_1h:J"]
```

```
Threshold direct: 24458.97
Threshold diffuse: 11822.505000000001
2599
```



[8]:
```
location  is_estimated
A         0                87
          1                10
B         0              1250
          1                32
C         0              1174
          1                46
Name: direct_rad_1h:J, dtype: int64
```

[9]:
```
# print num rows
temprows = len(X_train)
X_train.dropna(subset=['y', 'direct_rad_1h:J', 'diffuse_rad_1h:J'],␣
 ↪inplace=True)
print("Dropped rows: ", temprows - len(X_train))
```

```
Dropped rows:  1876
```

2.1.2 Other stuff

[10]:
```
import numpy as np
import pandas as pd

for attr in use_dt_attrs:
    X_train[attr] = getattr(X_train.index, attr)
    X_test[attr] = getattr(X_test.index, attr)

#print(X_train.head())


# If the "sample_weight" column is present and weight_evaluation is True,␣
 ↪multiply sample_weight with sample_weight_may_july if the ds is between␣
 ↪05-01 00:00:00 and 07-03 23:00:00, else add sample_weight as a column to␣
 ↪X_train
if weight_evaluation:
    if "sample_weight" not in X_train.columns:
        X_train["sample_weight"] = 1

    X_train.loc[((X_train.index.month >= 5) & (X_train.index.month <= 6)) |␣
 ↪((X_train.index.month == 7) & (X_train.index.day <= 3)), "sample_weight"] *=␣
 ↪sample_weight_may_july


print(X_train.iloc[200])
print(X_train[((X_train.index.month >= 5) & (X_train.index.month <= 6)) |␣
 ↪((X_train.index.month == 7) & (X_train.index.day <= 3))].head(1))
```

```
if use_groups:
    # fix groups for cross validation
    locations = X_train['location'].unique()  # Assuming 'location' is the name␣
 ↪of the column representing locations

    grouped_dfs = []  # To store data frames split by location

    # Loop through each unique location
    for loc in locations:
        loc_df = X_train[X_train['location'] == loc]

        # Sort the DataFrame for this location by the time column
        loc_df = loc_df.sort_index()

        # Calculate the size of each group for this location
        group_size = len(loc_df) // n_groups

        # Create a new 'group' column for this location
        loc_df['group'] = np.repeat(range(n_groups),␣
 ↪repeats=[group_size]*(n_groups-1) + [len(loc_df) - group_size*(n_groups-1)])

        # Append to list of grouped DataFrames
        grouped_dfs.append(loc_df)

    # Concatenate all the grouped DataFrames back together
    X_train = pd.concat(grouped_dfs)
    X_train.sort_index(inplace=True)
    print(X_train["group"].head())




X_train.drop(columns=to_drop, inplace=True)
X_test.drop(columns=to_drop, inplace=True)

X_train.to_csv('X_train_raw.csv', index=True)
X_test.to_csv('X_test_raw.csv', index=True)
```

```
absolute_humidity_2m:gm3          7.625
air_density_2m:kgm3              1.2215
ceiling_height_agl:m          3644.050049
clear_sky_energy_1h:J          2896336.75
clear_sky_rad:W                 753.849976
cloud_base_agl:m              3644.050049
dew_or_rime:idx                       0.0
```

```
dew_point_2m:K                    280.475006
diffuse_rad:W                     127.475006
diffuse_rad_1h:J                  526032.625
direct_rad:W                          488.0
direct_rad_1h:J                  1718048.625
effective_cloud_cover:p            18.200001
elevation:m                              6.0
fresh_snow_12h:cm                        0.0
fresh_snow_1h:cm                         0.0
fresh_snow_24h:cm                        0.0
fresh_snow_3h:cm                         0.0
fresh_snow_6h:cm                         0.0
is_day:idx                               1.0
is_in_shadow:idx                         0.0
msl_pressure:hPa                 1026.775024
precip_5min:mm                           0.0
precip_type_5min:idx                     0.0
pressure_100m:hPa                1013.599976
pressure_50m:hPa                 1019.599976
prob_rime:p                              0.0
rain_water:kgm2                          0.0
relative_humidity_1000hPa:p        53.825001
sfc_pressure:hPa                 1025.699951
snow_density:kgm3                        NaN
snow_depth:cm                            0.0
snow_drift:idx                           0.0
snow_melt_10min:mm                       0.0
snow_water:kgm2                          0.0
sun_azimuth:d                     222.089005
sun_elevation:d                    44.503498
super_cooled_liquid_water:kgm2           0.0
t_1000hPa:K                       286.700012
total_cloud_cover:p                18.200001
visibility:m                        52329.25
wind_speed_10m:ms                        2.6
wind_speed_u_10m:ms                     -1.9
wind_speed_v_10m:ms                    -1.75
wind_speed_w_1000hPa:ms                  0.0
is_estimated                               0
y                                     4367.44
location                                   A
Name: 2019-06-11 13:00:00, dtype: object
                     absolute_humidity_2m:gm3  air_density_2m:kgm3  \
ds
2019-06-02 23:00:00                       7.7                1.2235

                     ceiling_height_agl:m  clear_sky_energy_1h:J  \
ds
```

```
2019-06-02 23:00:00           1689.824951                    0.0

                     clear_sky_rad:W  cloud_base_agl:m  dew_or_rime:idx  \
ds
2019-06-02 23:00:00              0.0       1689.824951              0.0

                     dew_point_2m:K  diffuse_rad:W  diffuse_rad_1h:J  …  \
ds                                                                     …
2019-06-02 23:00:00      280.299988            0.0               0.0  …

                     t_1000hPa:K  total_cloud_cover:p  visibility:m  \
ds
2019-06-02 23:00:00   286.899994                100.0  33770.648438

                     wind_speed_10m:ms  wind_speed_u_10m:ms  \
ds
2019-06-02 23:00:00               3.35                -3.35

                     wind_speed_v_10m:ms  wind_speed_w_1000hPa:ms  \
ds
2019-06-02 23:00:00                0.275                      0.0

                     is_estimated    y  location
ds
2019-06-02 23:00:00             0  0.0         A

[1 rows x 48 columns]
```

[11]:
```
# Create a plot of X_train showing its "y" and color it based on the value of␣
↪the sample_weight column.
if "sample_weight" in X_train.columns:
    import matplotlib.pyplot as plt
    import seaborn as sns
    sns.scatterplot(data=X_train, x=X_train.index, y="y", hue="sample_weight",␣
↪palette="deep", size=3)
    plt.show()
```

[12]:
```
def normalize_sample_weights_per_location(df):
    for loc in locations:
        loc_df = df[df["location"] == loc]
        loc_df["sample_weight"] = loc_df["sample_weight"] /␣
↪loc_df["sample_weight"].sum() * loc_df.shape[0]
        df[df["location"] == loc] = loc_df
    return df

import pandas as pd
```

```python
def split_and_shuffle_data(input_data, num_bins, frac1):
    """
    Splits the input_data into num_bins and shuffles them, then divides the␣
 ↪bins into two datasets based on the given fraction for the first set.

    Args:
        input_data (pd.DataFrame): The data to be split and shuffled.
        num_bins (int): The number of bins to split the data into.
        frac1 (float): The fraction of each bin to go into the first output␣
 ↪dataset.

    Returns:
        pd.DataFrame, pd.DataFrame: The two output datasets.
    """
    # Validate the input fraction
    if frac1 < 0 or frac1 > 1:
        raise ValueError("frac1 must be between 0 and 1.")

    if frac1==1:
        return input_data, pd.DataFrame()

    # Calculate the fraction for the second output set
    frac2 = 1 - frac1

    # Calculate bin size
    bin_size = len(input_data) // num_bins

    # Initialize empty DataFrames for output
    output_data1 = pd.DataFrame()
    output_data2 = pd.DataFrame()

    for i in range(num_bins):
        # Shuffle the data in the current bin
        np.random.seed(i)
        current_bin = input_data.iloc[i * bin_size: (i + 1) * bin_size].
 ↪sample(frac=1)

        # Calculate the sizes for each output set
        size1 = int(len(current_bin) * frac1)

        # Split and append to output DataFrames
        output_data1 = pd.concat([output_data1, current_bin.iloc[:size1]])
        output_data2 = pd.concat([output_data2, current_bin.iloc[size1:]])

    # Shuffle and split the remaining data
    remaining_data = input_data.iloc[num_bins * bin_size:].sample(frac=1)
```

```
    remaining_size1 = int(len(remaining_data) * frac1)

    output_data1 = pd.concat([output_data1, remaining_data.iloc[:
 ↪remaining_size1]])
    output_data2 = pd.concat([output_data2, remaining_data.iloc[remaining_size1:
 ↪]])

    return output_data1, output_data2
```

[13]:
```
from autogluon.tabular import TabularDataset, TabularPredictor
data = TabularDataset('X_train_raw.csv')
# set group column of train_data be increasing from 0 to 7 based on time, the␣
 ↪first 1/8 of the data is group 0, the second 1/8 of the data is group 1, etc.
data['ds'] = pd.to_datetime(data['ds'])
data = data.sort_values(by='ds')

# # print size of the group for each location
# for loc in locations:
#     print(f"Location {loc}:")
#     print(train_data[train_data["location"] == loc].groupby('group').size())


# get end date of train data and subtract 3 months
#split_time = pd.to_datetime(train_data["ds"]).max() - pd.
 ↪Timedelta(hours=tune_and_test_length)
# 2022-10-28 22:00:00
split_time = pd.to_datetime("2022-10-28 22:00:00")
train_set = TabularDataset(data[data["ds"] < split_time])
estimated_set = TabularDataset(data[data["ds"] >= split_time]) # only estimated

test_set = pd.DataFrame()
tune_set = pd.DataFrame()
new_train_set = pd.DataFrame()

if not use_tune_data:
    raise Exception("Not implemented")

for location in locations:
    loc_data = data[data["location"] == location]
    num_train_rows = len(loc_data)

    tune_rows = 1500.0 # 2500.0
    if use_test_data:
        tune_rows = 1880.0#max(3000.0,␣
 ↪len(estimated_set[estimated_set["location"] == location]))
```

```
    holdout_frac = max(0.01, min(0.1, tune_rows / num_train_rows)) *␣
↪num_train_rows / len(estimated_set[estimated_set["location"] == location])

    print(f"Size of estimated for location {location}:␣
↪{len(estimated_set[estimated_set['location'] == location])}. Holdout frac␣
↪should be % of estimated: {holdout_frac}")

    # shuffle and split data
    loc_tune_set, loc_new_train_set =␣
↪split_and_shuffle_data(estimated_set[estimated_set['location'] == location],␣
↪40, holdout_frac)
    print(f"Length of location tune set : {len(loc_tune_set)}")
    new_train_set = pd.concat([new_train_set, loc_new_train_set])

    if use_test_data:
        loc_test_set, loc_tune_set = split_and_shuffle_data(loc_tune_set, 40, 0.
↪2)
        test_set = pd.concat([test_set, loc_test_set])

    tune_set = pd.concat([tune_set, loc_tune_set])




print("Length of train set before adding test set", len(train_set))
# add rest to train_set
train_set = pd.concat([train_set, new_train_set])
print("Length of train set after adding test set", len(train_set))




if use_groups:
    test_set = test_set.drop(columns=['group'])


tuning_data = tune_set

# number of rows in tuning data for each location
print("Shapes of tuning data", tuning_data.groupby('location').size())



if use_test_data:
    test_data = test_set
    print("Shape of test", test_data.shape[0])
```

```
train_data = train_set

# ensure sample weights for your training (or tuning) data sum to the number of␣
 ↪rows in the training (or tuning) data.
if weight_evaluation:
    # ensure sample weights for data sum to the number of rows in the tuning /
 ↪train data.
    tuning_data = normalize_sample_weights_per_location(tuning_data)
    train_data = normalize_sample_weights_per_location(train_data)
    if use_test_data:
        test_data = normalize_sample_weights_per_location(test_data)


train_data = TabularDataset(train_data)
tuning_data = TabularDataset(tuning_data)

if use_test_data:
    test_data = TabularDataset(test_data)
```

```
Size of estimated for location A: 4214. Holdout frac should be % of estimated:
0.4461319411485524
Length of location tune set : 1846
Size of estimated for location B: 3533. Holdout frac should be % of estimated:
0.5321256722332296
Length of location tune set : 1846
Size of estimated for location C: 2923. Holdout frac should be % of estimated:
0.6431748203900103
Length of location tune set : 1841
Length of train set before adding test set 77247
Length of train set after adding test set 82384
Shapes of tuning data location
A    1485
B    1485
C    1481
dtype: int64
Shape of test 1082
```

3 Quick EDA

[14]:
```
if run_analysis:
    import autogluon.eda.auto as auto
    auto.dataset_overview(train_data=train_data, test_data=test_data,␣
 ↪label="y", sample=None)
```

[15]:
```
if run_analysis:
    auto.target_analysis(train_data=train_data, label="y", sample=None)
```

4 Modeling

[16]:
```
import os


# Get the last submission number
last_submission_number = int(max([int(filename.split('_')[1].split('.')[0]) for␣
 ↪filename in os.listdir('submissions') if "submission" in filename]))
print("Last submission number:", last_submission_number)
print("Now creating submission number:", last_submission_number + 1)

# Create the new filename
new_filename = f'submission_{last_submission_number + 1}'

hello = os.environ.get('HELLO')
if hello is not None:
    new_filename += f'_{hello}'

print("New filename:", new_filename)
```

```
Last submission number: 121
Now creating submission number: 122
New filename: submission_122
```

[17]:
```
predictors = [None, None, None]
```

[18]:
```
def fit_predictor_for_location(loc):
    print(f"Training model for location {loc}...")
    # sum of sample weights for this location, and number of rows, for both␣
 ↪train and tune data and test data
    if weight_evaluation:
        print("Train data sample weight sum:",␣
 ↪train_data[train_data["location"] == loc]["sample_weight"].sum())
        print("Train data number of rows:", train_data[train_data["location"]␣
 ↪== loc].shape[0])
        if use_tune_data:
            print("Tune data sample weight sum:",␣
 ↪tuning_data[tuning_data["location"] == loc]["sample_weight"].sum())
            print("Tune data number of rows:",␣
 ↪tuning_data[tuning_data["location"] == loc].shape[0])
        if use_test_data:
            print("Test data sample weight sum:",␣
 ↪test_data[test_data["location"] == loc]["sample_weight"].sum())
            print("Test data number of rows:", test_data[test_data["location"]␣
 ↪== loc].shape[0])
    predictor = TabularPredictor(
        label=label,
```

```
        eval_metric=metric,
        path=f"AutogluonModels/{new_filename}_{loc}",
        # sample_weight=sample_weight,
        # weight_evaluation=weight_evaluation,
        # groups="group" if use_groups else None,
    ).fit(
        train_data=train_data[train_data["location"] == loc].
↪drop(columns=["ds"]),
        time_limit=time_limit,
        presets=presets,
        num_stack_levels=num_stack_levels,
        num_bag_folds=num_bag_folds if not use_groups else 2,# just put␣
↪somethin, will be overwritten anyways
        num_bag_sets=num_bag_sets,
        tuning_data=tuning_data[tuning_data["location"] == loc].
↪reset_index(drop=True).drop(columns=["ds"]) if use_tune_data else None,
        use_bag_holdout=use_bag_holdout,
        # holdout_frac=holdout_frac,
        excluded_model_types=excluded_model_types
    )

    # evaluate on test data
    if use_test_data:
        # drop sample_weight column
        t = test_data[test_data["location"] == loc]#.
↪drop(columns=["sample_weight"])
        perf = predictor.evaluate(t)
        print("Evaluation on test data:")
        print(perf[predictor.eval_metric.name])

    return predictor

loc = "A"
predictors[0] = fit_predictor_for_location(loc)
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_122_A"
Presets specified: ['experimental_zeroshot_hpo_hybrid']
Stack configuration (auto_stack=True): num_stack_levels=0, num_bag_folds=8,
num_bag_sets=20
Beginning AutoGluon training … Time limit = 3600s
AutoGluon will save models to "AutogluonModels/submission_122_A/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Linux
Platform Machine:   x86_64
Platform Version:   #1 SMP Debian 5.10.197-1 (2023-09-29)
```

```
Disk Space Avail:   155.32 GB / 315.93 GB (49.2%)
Train Data Rows:    30934
Train Data Columns: 44
Tuning Data Rows:    1485
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (5733.42, 0.0, 673.41535, 1195.24)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    132339.61 MB
        Train Data (Original)  Memory Usage: 13.03 MB (0.0% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.

Training model for location A…

        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
```

```
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.2s = Fit runtime
        41 features in original data used to generate 41 features in processed
data.
        Train Data (Processed) Memory Usage: 10.18 MB (0.0% of available memory)
Data preprocessing and feature engineering runtime = 0.18s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}, {'min_samples_leaf': 1,
'max_leaf_nodes': 15000, 'max_features': 0.5, 'ag_args': {'name_suffix': '_r19',
'priority': 20}}],
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}, {'min_samples_leaf': 5,
'max_leaf_nodes': 50000, 'max_features': 0.5, 'ag_args': {'name_suffix': '_r5',
'priority': 19}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge', {'extra_trees': False, 'feature_fraction': 0.7248284762542815,
'learning_rate': 0.07947286942946127, 'min_data_in_leaf': 50, 'num_leaves': 89,
'ag_args': {'name_suffix': '_r158', 'priority': 18}}, {'extra_trees': True,
'feature_fraction': 0.7832570544199176, 'learning_rate': 0.021720607471727896,
'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args': {'name_suffix': '_r118',
'priority': 17}}, {'extra_trees': True, 'feature_fraction': 0.7113010892989156,
'learning_rate': 0.012535427424259274, 'min_data_in_leaf': 16, 'num_leaves': 48,
'ag_args': {'name_suffix': '_r97', 'priority': 16}}, {'extra_trees': True,
'feature_fraction': 0.45555769907110816, 'learning_rate': 0.009591347321206594,
'min_data_in_leaf': 50, 'num_leaves': 110, 'ag_args': {'name_suffix': '_r71',
'priority': 15}}, {'extra_trees': False, 'feature_fraction':
0.40979710161022476, 'learning_rate': 0.008708890211023034, 'min_data_in_leaf':
3, 'num_leaves': 80, 'ag_args': {'name_suffix': '_r111', 'priority': 14}}],
        'XGB': {},
        'FASTAI': [{}, {'bs': 1024, 'emb_drop': 0.6167722379778131, 'epochs':
44, 'layers': [200, 100, 50], 'lr': 0.053440377855629266, 'ps':
```

```
0.48477211305443607, 'ag_args': {'name_suffix': '_r25', 'priority': 13}}, {'bs':
1024, 'emb_drop': 0.6046989241462619, 'epochs': 48, 'layers': [200, 100, 50],
'lr': 0.00775309042164966, 'ps': 0.09244767444160731, 'ag_args': {'name_suffix':
'_r51', 'priority': 12}}, {'bs': 512, 'emb_drop': 0.6557225316526186, 'epochs':
49, 'layers': [200, 100], 'lr': 0.023627682025564638, 'ps': 0.519566584552178,
'ag_args': {'name_suffix': '_r82', 'priority': 11}}, {'bs': 2048, 'emb_drop':
0.4066210919034579, 'epochs': 43, 'layers': [400, 200], 'lr':
0.0029598312717673434, 'ps': 0.4378695797438974, 'ag_args': {'name_suffix':
'_r121', 'priority': 10}}, {'bs': 128, 'emb_drop': 0.44339037504795686,
'epochs': 31, 'layers': [400, 200, 100], 'lr': 0.008615195908919904, 'ps':
0.19220253419114286, 'ag_args': {'name_suffix': '_r145', 'priority': 9}}, {'bs':
128, 'emb_drop': 0.12106594798980945, 'epochs': 38, 'layers': [200, 100, 50],
'lr': 0.037991970245029975, 'ps': 0.33120008492595093, 'ag_args':
{'name_suffix': '_r173', 'priority': 8}}, {'bs': 128, 'emb_drop':
0.4599138419358, 'epochs': 47, 'layers': [200, 100], 'lr': 0.03888383281136287,
'ps': 0.28193673177122863, 'ag_args': {'name_suffix': '_r128', 'priority': 7}}],
        'CAT': [{}, {'depth': 5, 'l2_leaf_reg': 4.774992314058497,
'learning_rate': 0.038551267822920274, 'ag_args': {'name_suffix': '_r16',
'priority': 6}}, {'depth': 4, 'l2_leaf_reg': 1.9950125740798321,
'learning_rate': 0.028091050379971633, 'ag_args': {'name_suffix': '_r42',
'priority': 5}}, {'depth': 6, 'l2_leaf_reg': 1.8298803017644376,
'learning_rate': 0.017844259810823604, 'ag_args': {'name_suffix': '_r93',
'priority': 4}}, {'depth': 7, 'l2_leaf_reg': 4.81099604606794, 'learning_rate':
0.019085060180573103, 'ag_args': {'name_suffix': '_r44', 'priority': 3}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['RF', 'XGB', 'CAT'] (Specified by `excluded_model_types`)
Fitting 21 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 3599.82s of
the 3599.82s of remaining time.
        -191.231         = Validation score   (-mean_absolute_error)
        0.04s    = Training   runtime
        1.25s    = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 3598.25s of
the 3598.25s of remaining time.
        -192.9182        = Validation score   (-mean_absolute_error)
        0.04s    = Training   runtime
        0.4s     = Validation runtime
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 3597.65s of the
3597.65s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -85.9426         = Validation score   (-mean_absolute_error)
        33.25s   = Training   runtime
        14.42s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 3554.93s of the
3554.93s of remaining time.
```

```
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -90.5139        = Validation score   (-mean_absolute_error)
        30.39s   = Training   runtime
        7.49s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 … Training model for up to 3520.37s of the
3520.37s of remaining time.
        -102.5531       = Validation score   (-mean_absolute_error)
        1.81s    = Training   runtime
        1.1s     = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 3515.25s of
the 3515.24s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -103.8453       = Validation score   (-mean_absolute_error)
        38.06s   = Training   runtime
        0.49s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 3474.52s of
the 3474.52s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -88.0553        = Validation score   (-mean_absolute_error)
        99.41s   = Training   runtime
        0.36s    = Validation runtime
Fitting model: ExtraTrees_r19_BAG_L1 … Training model for up to 3373.83s of
the 3373.83s of remaining time.
        -100.9585       = Validation score   (-mean_absolute_error)
        1.13s    = Training   runtime
        1.11s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 3368.96s of the
3368.96s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -92.787  = Validation score   (-mean_absolute_error)
        49.14s   = Training   runtime
        12.84s   = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 3311.47s of the
3311.46s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -85.8912        = Validation score   (-mean_absolute_error)
        21.26s   = Training   runtime
        18.18s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 3285.14s of the
3285.14s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -86.3739        = Validation score   (-mean_absolute_error)
```

```
        34.3s    = Training   runtime
        25.28s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 3243.85s of the
3243.84s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -92.7495         = Validation score   (-mean_absolute_error)
        82.06s   = Training   runtime
        49.4s    = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 3148.57s of the
3148.57s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -88.6662         = Validation score   (-mean_absolute_error)
        55.43s   = Training   runtime
        26.39s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 3082.63s
of the 3082.63s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -104.2684        = Validation score   (-mean_absolute_error)
        23.48s   = Training   runtime
        0.24s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 3057.8s
of the 3057.79s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -97.1141         = Validation score   (-mean_absolute_error)
        25.58s   = Training   runtime
        0.24s    = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 3030.67s
of the 3030.66s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -106.6312        = Validation score   (-mean_absolute_error)
        35.99s   = Training   runtime
        0.31s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 2992.84s
of the 2992.84s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -117.7474        = Validation score   (-mean_absolute_error)
        20.56s   = Training   runtime
        0.22s    = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 2970.6s
of the 2970.59s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
```

```
        -96.1916        = Validation score   (-mean_absolute_error)
        88.24s   = Training   runtime
        0.89s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 2880.42s
of the 2880.41s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -100.9066       = Validation score   (-mean_absolute_error)
        93.2s    = Training   runtime
        0.84s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 2785.23s
of the 2785.22s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -102.5638       = Validation score   (-mean_absolute_error)
        105.23s  = Training   runtime
        0.8s     = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 2678.0s of the
2678.0s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -87.4068        = Validation score   (-mean_absolute_error)
        107.91s  = Training   runtime
        24.4s    = Validation runtime
Repeating k-fold bagging: 2/20
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 2560.22s of the
2560.21s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -86.1803        = Validation score   (-mean_absolute_error)
        67.87s   = Training   runtime
        33.59s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 2519.62s of the
2519.62s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -90.3028        = Validation score   (-mean_absolute_error)
        63.1s    = Training   runtime
        14.2s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 2482.13s of
the 2482.13s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -103.0736       = Validation score   (-mean_absolute_error)
        76.68s   = Training   runtime
        1.05s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 2441.65s of
the 2441.65s of remaining time.
```

```
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -86.6093        = Validation score   (-mean_absolute_error)
        208.19s  = Training   runtime
        0.73s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 2330.96s of the
2330.95s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -91.3734        = Validation score   (-mean_absolute_error)
        92.47s   = Training   runtime
        19.27s   = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 2280.33s of the
2280.32s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -85.5189        = Validation score   (-mean_absolute_error)
        45.19s   = Training   runtime
        27.78s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 2251.74s of the
2251.74s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -86.5744        = Validation score   (-mean_absolute_error)
        73.55s   = Training   runtime
        47.53s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 2205.36s of the
2205.36s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -92.7942        = Validation score   (-mean_absolute_error)
        160.78s  = Training   runtime
        98.26s   = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 2108.48s of the
2108.48s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -88.7043        = Validation score   (-mean_absolute_error)
        111.6s   = Training   runtime
        48.38s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 2041.96s
of the 2041.96s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -104.3129       = Validation score   (-mean_absolute_error)
        47.03s   = Training   runtime
        0.47s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 2016.79s
```

```
of the 2016.79s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -95.9834        = Validation score   (-mean_absolute_error)
        50.65s   = Training   runtime
        0.46s    = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 1989.9s
of the 1989.9s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -106.5279       = Validation score   (-mean_absolute_error)
        72.81s   = Training   runtime
        0.63s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 1951.35s
of the 1951.34s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -118.3212       = Validation score   (-mean_absolute_error)
        40.71s   = Training   runtime
        0.46s    = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 1929.69s
of the 1929.69s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -95.9189        = Validation score   (-mean_absolute_error)
        177.31s  = Training   runtime
        1.79s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 1838.01s
of the 1838.0s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -99.3103        = Validation score   (-mean_absolute_error)
        187.28s  = Training   runtime
        1.67s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 1741.46s
of the 1741.46s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -104.2124       = Validation score   (-mean_absolute_error)
        207.72s  = Training   runtime
        1.6s     = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 1636.48s of the
1636.48s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -87.1874        = Validation score   (-mean_absolute_error)
        218.28s  = Training   runtime
        47.47s   = Validation runtime
```

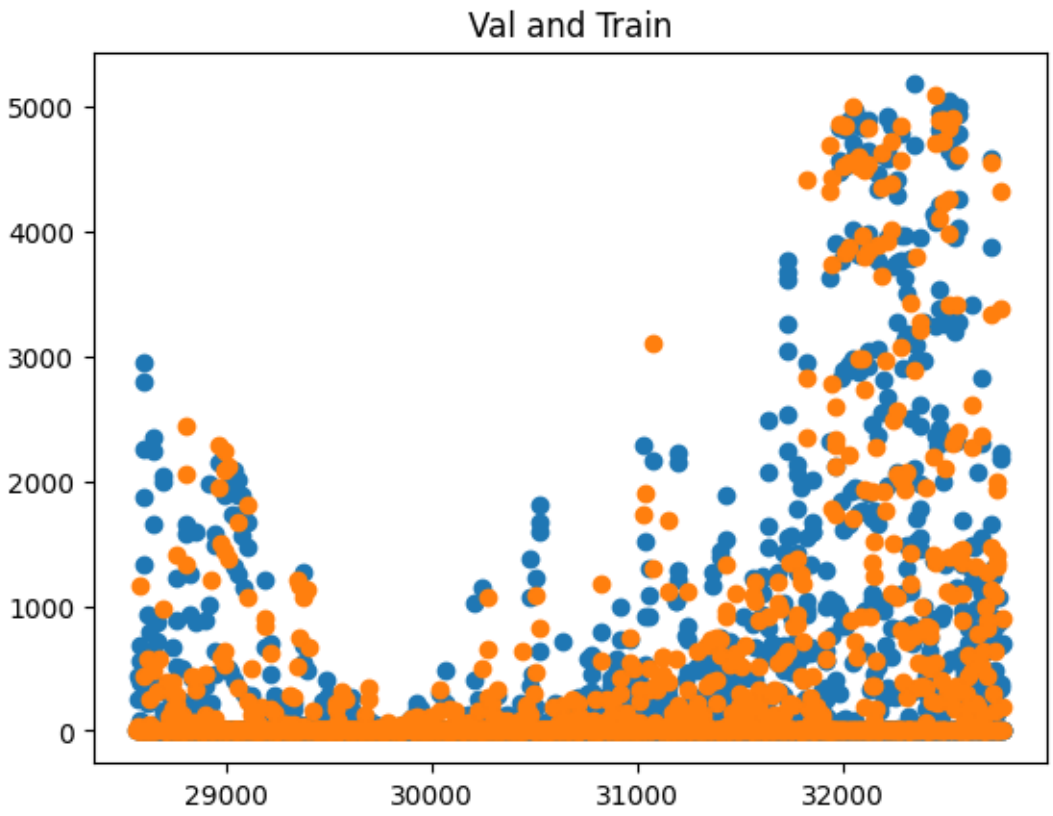
```
Repeating k-fold bagging: 3/20
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 1512.34s of the
1512.34s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -86.1411         = Validation score   (-mean_absolute_error)
        102.14s  = Training   runtime
        47.86s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 1471.25s of the
1471.25s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -90.3661         = Validation score   (-mean_absolute_error)
        93.52s   = Training   runtime
        20.03s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 1435.42s of
the 1435.41s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -102.877         = Validation score   (-mean_absolute_error)
        116.2s   = Training   runtime
        1.54s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 1393.7s of the
1393.69s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -86.9808         = Validation score   (-mean_absolute_error)
        319.28s  = Training   runtime
        1.11s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 1280.64s of the
1280.64s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -90.9068         = Validation score   (-mean_absolute_error)
        134.71s  = Training   runtime
        28.03s   = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 1228.1s of the
1228.1s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -85.2599         = Validation score   (-mean_absolute_error)
        68.73s   = Training   runtime
        38.31s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 1199.12s of the
1199.12s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -86.5317         = Validation score   (-mean_absolute_error)
```

```
        114.17s  = Training   runtime
        71.39s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 1150.13s of the
1150.13s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -92.6296         = Validation score   (-mean_absolute_error)
        238.69s  = Training   runtime
        139.59s  = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 1050.86s of the
1050.85s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -88.7444         = Validation score   (-mean_absolute_error)
        167.53s  = Training   runtime
        71.01s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 982.17s
of the 982.16s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -103.6838        = Validation score   (-mean_absolute_error)
        70.2s    = Training   runtime
        0.71s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 957.32s
of the 957.31s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -95.0394         = Validation score   (-mean_absolute_error)
        76.12s   = Training   runtime
        0.7s     = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 929.97s
of the 929.97s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -107.3447        = Validation score   (-mean_absolute_error)
        108.83s  = Training   runtime
        0.96s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 891.9s
of the 891.89s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -119.2008        = Validation score   (-mean_absolute_error)
        60.77s   = Training   runtime
        0.69s    = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 870.22s
of the 870.21s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
```

```
        -95.1641         = Validation score   (-mean_absolute_error)
        265.62s  = Training   runtime
        2.7s     = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 778.8s
of the 778.8s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -98.4686         = Validation score   (-mean_absolute_error)
        280.87s  = Training   runtime
        2.48s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 682.13s
of the 682.13s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -103.41  = Validation score   (-mean_absolute_error)
        311.95s  = Training   runtime
        2.45s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 574.81s of the
574.81s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -87.2987         = Validation score   (-mean_absolute_error)
        326.89s  = Training   runtime
        70.2s    = Validation runtime
Completed 3/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 360.0s of the
448.79s of remaining time.
        -82.2346         = Validation score   (-mean_absolute_error)
        0.53s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3151.77s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_122_A/")
Evaluation: mean_absolute_error on test data: -105.21386474786952
        Note: Scores are always higher_is_better. This metric score can be
multiplied by -1 to get the metric value.
Evaluations on test data:
{
    "mean_absolute_error": -105.21386474786952,
    "root_mean_squared_error": -334.6204247772324,
    "mean_squared_error": -111970.82867809547,
    "r2": 0.8242712281113946,
    "pearsonr": 0.9122558218574904,
    "median_absolute_error": -2.772814989089966
}

Evaluation on test data:
```

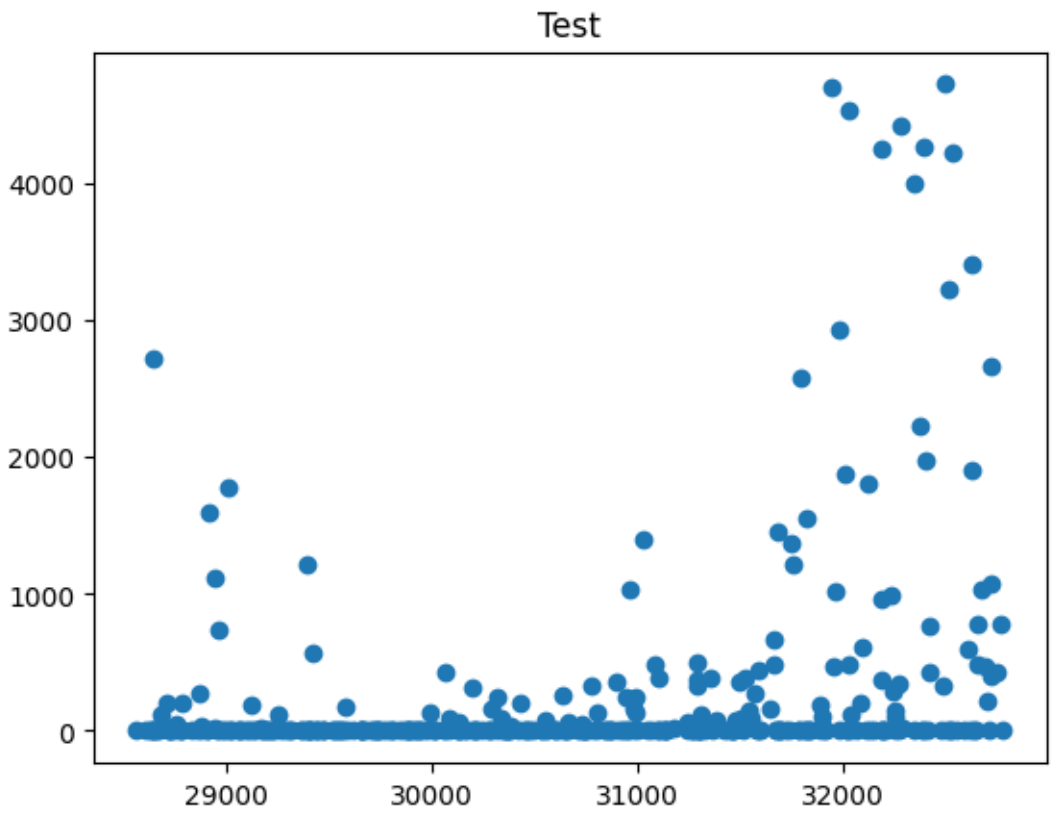
-105.21386474786952

[19]:
```
import matplotlib.pyplot as plt
leaderboards = [None, None, None]
def leaderboard_for_location(i, loc):
    if use_tune_data:
        plt.scatter(train_data[(train_data["location"] == loc) &␣
 ↪(train_data["is_estimated"]==True)]["y"].index,␣
 ↪train_data[(train_data["location"] == loc) &␣
 ↪(train_data["is_estimated"]==True)]["y"])
        plt.scatter(tuning_data[tuning_data["location"] == loc]["y"].index,␣
 ↪tuning_data[tuning_data["location"] == loc]["y"])
        plt.title("Val and Train")
        plt.show()

        if use_test_data:
            lb = predictors[i].leaderboard(test_data[test_data["location"] ==␣
 ↪loc])
            lb["location"] = loc
            plt.scatter(test_data[test_data["location"] == loc]["y"].index,␣
 ↪test_data[test_data["location"] == loc]["y"])
            plt.title("Test")

            return lb

    return pd.DataFrame()

leaderboards[0] = leaderboard_for_location(0, loc)
```




```
                        model  score_test   score_val  pred_time_test
pred_time_val    fit_time  pred_time_test_marginal  pred_time_val_marginal
fit_time_marginal  stack_level  can_infer  fit_order
0           WeightedEnsemble_L2 -105.213865  -82.234560       15.416068
158.172757  893.704767                0.004306               0.000686
0.533880            2       True         22
1             LightGBMXT_BAG_L1 -105.651709  -86.141116        2.763916
47.861973  102.143539                2.763916              47.861973
102.143539            1       True          3
2    NeuralNetFastAI_r51_BAG_L1 -106.655413  -95.039358        0.376366
0.700547   76.118525                0.376366               0.700547
76.118525            1       True         15
3          LightGBM_r118_BAG_L1 -106.870157  -85.259908        2.344336
38.308059   68.731673                2.344336              38.308059
68.731673            1       True         10
4           LightGBM_r97_BAG_L1 -107.586386  -86.531742        4.333927
71.390422  114.166910                4.333927              71.390422
114.166910            1       True         11
5         NeuralNetTorch_BAG_L1 -112.454177  -86.980764        0.569817
1.105011  319.282644                0.569817               1.105011
319.282644            1       True          7
```

```
6           LightGBM_r71_BAG_L1 -114.721642  -92.629558        8.428360
139.588237  238.691140                 8.428360              139.588237
238.691140           1       True         12
7          LightGBM_r111_BAG_L1 -115.662826  -88.744429        6.469665
71.013778  167.530970                 6.469665               71.013778
167.530970           1       True         13
8   NeuralNetFastAI_r173_BAG_L1 -116.151949  -98.468557        0.874867
2.481646  280.872100                 0.874867                2.481646
280.872100           1       True         19
9          LightGBMLarge_BAG_L1 -117.513573  -87.298706        9.357328
70.196481  326.894507                 9.357328               70.196481
326.894507           1       True         21
10  NeuralNetFastAI_r145_BAG_L1 -117.535234  -95.164051        0.815387
2.704688  265.617656                 0.815387                2.704688
265.617656           1       True         18
11              LightGBM_BAG_L1 -118.126742  -90.366088        2.076667
20.028206   93.519575                 2.076667               20.028206
93.519575           1       True          4
12       NeuralNetFastAI_BAG_L1 -120.371544 -102.877024        0.527273
1.540404  116.199956                 0.527273                1.540404
116.199956           1       True          6
13   NeuralNetFastAI_r25_BAG_L1 -120.402365 -103.683760        0.377176
0.710119   70.197967                 0.377176                0.710119
70.197967           1       True         14
14         LightGBM_r158_BAG_L1 -120.484917  -90.906798        2.837722
28.030752  134.713080                 2.837722               28.030752
134.713080           1       True          9
15  NeuralNetFastAI_r128_BAG_L1 -122.006312 -103.410039        1.118670
2.450946  311.951262                 1.118670                2.450946
311.951262           1       True         20
16        ExtraTrees_r19_BAG_L1 -126.033456 -100.958547        0.556135
1.114801    1.126558                 0.556135                1.114801
1.126558           1       True          8
17   NeuralNetFastAI_r82_BAG_L1 -127.562204 -107.344693        0.419232
0.961000  108.827765                 0.419232                0.961000
108.827765           1       True         16
18         ExtraTreesMSE_BAG_L1 -130.386831 -102.553116        0.557890
1.103508    1.808400                 0.557890                1.103508
1.808400           1       True          5
19  NeuralNetFastAI_r121_BAG_L1 -139.844186 -119.200840        0.314491
0.688419   60.771383                 0.314491                0.688419
60.771383           1       True         17
20        KNeighborsDist_BAG_L1 -189.567130 -192.918160        0.013305
0.397008    0.035783                 0.013305                0.397008
0.035783           1       True          2
21        KNeighborsUnif_BAG_L1 -191.283846 -191.231007        0.143528
1.246812    0.037004                 0.143528                1.246812
0.037004           1       True          1
```



[20]:
```
loc = "B"
predictors[1] = fit_predictor_for_location(loc)
leaderboards[1] = leaderboard_for_location(1, loc)
```

```
Presets specified: ['experimental_zeroshot_hpo_hybrid']
Stack configuration (auto_stack=True): num_stack_levels=0, num_bag_folds=8,
num_bag_sets=20
Beginning AutoGluon training … Time limit = 3600s
AutoGluon will save models to "AutogluonModels/submission_122_B/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Linux
Platform Machine:   x86_64
Platform Version:   #1 SMP Debian 5.10.197-1 (2023-09-29)
Disk Space Avail:   147.52 GB / 315.93 GB (46.7%)
Train Data Rows:    27377
Train Data Columns: 44
Tuning Data Rows:    1485
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
```

```
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (1152.3, -0.0, 98.11625, 206.48535)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …

Training model for location B…

Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    130392.4 MB
        Train Data (Original)  Memory Usage: 11.6 MB (0.0% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 2): ['elevation:m', 'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 41 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.2s = Fit runtime
        42 features in original data used to generate 42 features in processed
data.
        Train Data (Processed) Memory Usage: 9.29 MB (0.0% of available memory)
Data preprocessing and feature engineering runtime = 0.22s …
AutoGluon will gauge predictive performance using evaluation metric:
```

```
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}, {'min_samples_leaf': 1,
'max_leaf_nodes': 15000, 'max_features': 0.5, 'ag_args': {'name_suffix': '_r19',
'priority': 20}}],
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}, {'min_samples_leaf': 5,
'max_leaf_nodes': 50000, 'max_features': 0.5, 'ag_args': {'name_suffix': '_r5',
'priority': 19}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge', {'extra_trees': False, 'feature_fraction': 0.7248284762542815,
'learning_rate': 0.07947286942946127, 'min_data_in_leaf': 50, 'num_leaves': 89,
'ag_args': {'name_suffix': '_r158', 'priority': 18}}, {'extra_trees': True,
'feature_fraction': 0.7832570544199176, 'learning_rate': 0.021720607471727896,
'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args': {'name_suffix': '_r118',
'priority': 17}}, {'extra_trees': True, 'feature_fraction': 0.7113010892989156,
'learning_rate': 0.012535427424259274, 'min_data_in_leaf': 16, 'num_leaves': 48,
'ag_args': {'name_suffix': '_r97', 'priority': 16}}, {'extra_trees': True,
'feature_fraction': 0.45555769907110816, 'learning_rate': 0.009591347321206594,
'min_data_in_leaf': 50, 'num_leaves': 110, 'ag_args': {'name_suffix': '_r71',
'priority': 15}}, {'extra_trees': False, 'feature_fraction':
0.40979710161022476, 'learning_rate': 0.008708890211023034, 'min_data_in_leaf':
3, 'num_leaves': 80, 'ag_args': {'name_suffix': '_r111', 'priority': 14}}],
        'XGB': {},
        'FASTAI': [{}, {'bs': 1024, 'emb_drop': 0.6167722379778131, 'epochs':
44, 'layers': [200, 100, 50], 'lr': 0.053440377855629266, 'ps':
0.48477211305443607, 'ag_args': {'name_suffix': '_r25', 'priority': 13}}, {'bs':
1024, 'emb_drop': 0.6046989241462619, 'epochs': 48, 'layers': [200, 100, 50],
'lr': 0.00775309042164966, 'ps': 0.09244767444160731, 'ag_args': {'name_suffix':
'_r51', 'priority': 12}}, {'bs': 512, 'emb_drop': 0.6557225316526186, 'epochs':
49, 'layers': [200, 100], 'lr': 0.023627682025564638, 'ps': 0.519566584552178,
'ag_args': {'name_suffix': '_r82', 'priority': 11}}, {'bs': 2048, 'emb_drop':
0.4066210919034579, 'epochs': 43, 'layers': [400, 200], 'lr':
0.0029598312717673434, 'ps': 0.4378695797438974, 'ag_args': {'name_suffix':
```

```
'_r121', 'priority': 10}}, {'bs': 128, 'emb_drop': 0.44339037504795686,
'epochs': 31, 'layers': [400, 200, 100], 'lr': 0.008615195908919904, 'ps':
0.19220253419114286, 'ag_args': {'name_suffix': '_r145', 'priority': 9}}, {'bs':
128, 'emb_drop': 0.12106594798980945, 'epochs': 38, 'layers': [200, 100, 50],
'lr': 0.037991970245029975, 'ps': 0.33120008492595093, 'ag_args':
{'name_suffix': '_r173', 'priority': 8}}, {'bs': 128, 'emb_drop':
0.4599138419358, 'epochs': 47, 'layers': [200, 100], 'lr': 0.03888383281136287,
'ps': 0.28193673177122863, 'ag_args': {'name_suffix': '_r128', 'priority': 7}}],
        'CAT': [{}, {'depth': 5, 'l2_leaf_reg': 4.774992314058497,
'learning_rate': 0.038551267822920274, 'ag_args': {'name_suffix': '_r16',
'priority': 6}}, {'depth': 4, 'l2_leaf_reg': 1.9950125740798321,
'learning_rate': 0.028091050379971633, 'ag_args': {'name_suffix': '_r42',
'priority': 5}}, {'depth': 6, 'l2_leaf_reg': 1.8298803017644376,
'learning_rate': 0.017844259810823604, 'ag_args': {'name_suffix': '_r93',
'priority': 4}}, {'depth': 7, 'l2_leaf_reg': 4.81099604606794, 'learning_rate':
0.019085060180573103, 'ag_args': {'name_suffix': '_r44', 'priority': 3}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['RF', 'XGB', 'CAT'] (Specified by `excluded_model_types`)
Fitting 21 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 3599.78s of
the 3599.77s of remaining time.
        -28.5444         = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.39s    = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 3599.27s of
the 3599.27s of remaining time.
        -28.798  = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.43s    = Validation runtime
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 3598.74s of the
3598.74s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.5683         = Validation score   (-mean_absolute_error)
        31.51s   = Training   runtime
        15.52s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 3562.06s of the
3562.05s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.6686         = Validation score   (-mean_absolute_error)
        35.32s   = Training   runtime
        16.27s   = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 … Training model for up to 3522.15s of the
3522.15s of remaining time.
        -15.393  = Validation score   (-mean_absolute_error)
```

```
        1.49s    = Training   runtime
        0.91s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 3518.87s of
the 3518.87s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.3296         = Validation score   (-mean_absolute_error)
        35.63s   = Training   runtime
        0.44s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 3481.59s of
the 3481.59s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8806         = Validation score   (-mean_absolute_error)
        146.46s  = Training   runtime
        0.37s    = Validation runtime
Fitting model: ExtraTrees_r19_BAG_L1 … Training model for up to 3333.53s of
the 3333.53s of remaining time.
        -15.2571         = Validation score   (-mean_absolute_error)
        0.97s    = Training   runtime
        0.92s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 3330.74s of the
3330.73s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -15.1218         = Validation score   (-mean_absolute_error)
        45.04s   = Training   runtime
        10.0s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 3277.52s of the
3277.52s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8851         = Validation score   (-mean_absolute_error)
        21.08s   = Training   runtime
        10.15s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 3252.52s of the
3252.51s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4763         = Validation score   (-mean_absolute_error)
        38.86s   = Training   runtime
        20.27s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 3208.16s of the
3208.16s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.8568         = Validation score   (-mean_absolute_error)
        76.57s   = Training   runtime
```

```
        43.67s   = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 3118.5s of the
3118.49s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4808         = Validation score   (-mean_absolute_error)
        53.46s   = Training   runtime
        20.25s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 3057.06s
of the 3057.05s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.6085         = Validation score   (-mean_absolute_error)
        21.18s   = Training   runtime
        0.22s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 3034.53s
of the 3034.53s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4434         = Validation score   (-mean_absolute_error)
        23.12s   = Training   runtime
        0.21s    = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 3009.89s
of the 3009.88s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.5635         = Validation score   (-mean_absolute_error)
        31.9s    = Training   runtime
        0.27s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 2976.35s
of the 2976.34s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.5206         = Validation score   (-mean_absolute_error)
        17.71s   = Training   runtime
        0.2s     = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 2957.09s
of the 2957.09s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.1404         = Validation score   (-mean_absolute_error)
        78.14s   = Training   runtime
        0.86s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 2876.97s
of the 2876.96s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.5364         = Validation score   (-mean_absolute_error)
```

```
        80.21s   = Training   runtime
        0.82s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 2794.86s
of the 2794.86s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.4378         = Validation score   (-mean_absolute_error)
        88.95s   = Training   runtime
        0.83s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 2704.0s of the
2704.0s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.1201         = Validation score   (-mean_absolute_error)
        107.96s  = Training   runtime
        20.12s   = Validation runtime
Repeating k-fold bagging: 2/20
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 2586.48s of the
2586.48s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4977         = Validation score   (-mean_absolute_error)
        61.58s   = Training   runtime
        28.91s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 2550.59s of the
2550.59s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.5685         = Validation score   (-mean_absolute_error)
        69.39s   = Training   runtime
        27.32s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 2510.69s of
the 2510.68s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.2696         = Validation score   (-mean_absolute_error)
        70.5s    = Training   runtime
        0.88s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 2473.96s of
the 2473.96s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8787         = Validation score   (-mean_absolute_error)
        261.2s   = Training   runtime
        0.75s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 2357.56s of the
2357.56s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
```

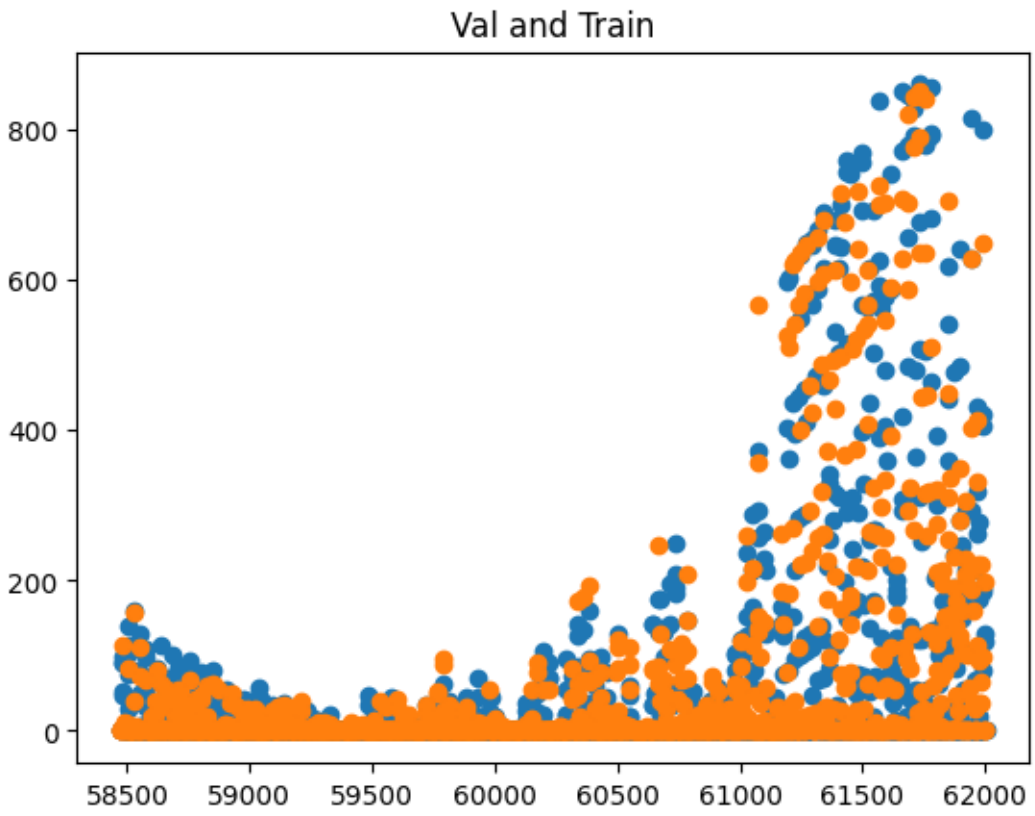
```
ParallelLocalFoldFittingStrategy
        -14.9927        = Validation score   (-mean_absolute_error)
        99.8s    = Training   runtime
        22.26s   = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 2292.79s of the
2292.78s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.9312        = Validation score   (-mean_absolute_error)
        42.74s   = Training   runtime
        22.36s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 2266.71s of the
2266.7s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4148        = Validation score   (-mean_absolute_error)
        77.57s   = Training   runtime
        40.21s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 2220.71s of the
2220.7s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.8498        = Validation score   (-mean_absolute_error)
        152.56s  = Training   runtime
        90.59s   = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 2126.41s of the
2126.4s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.489  = Validation score   (-mean_absolute_error)
        106.84s  = Training   runtime
        40.05s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 2061.96s
of the 2061.95s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.2517        = Validation score   (-mean_absolute_error)
        42.33s   = Training   runtime
        0.44s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 2039.29s
of the 2039.29s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.3601        = Validation score   (-mean_absolute_error)
        45.84s   = Training   runtime
        0.46s    = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 2014.78s
of the 2014.77s of remaining time.
```

```
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.5315        = Validation score   (-mean_absolute_error)
        64.49s   = Training   runtime
        0.57s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 1980.57s
of the 1980.56s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.5741        = Validation score   (-mean_absolute_error)
        35.36s   = Training   runtime
        0.41s    = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 1961.35s
of the 1961.34s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.2858        = Validation score   (-mean_absolute_error)
        157.01s  = Training   runtime
        1.65s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 1879.95s
of the 1879.95s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.911  = Validation score   (-mean_absolute_error)
        158.12s  = Training   runtime
        1.56s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 1799.66s
of the 1799.66s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.2643        = Validation score   (-mean_absolute_error)
        173.77s  = Training   runtime
        1.52s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 1712.49s of the
1712.48s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.0138        = Validation score   (-mean_absolute_error)
        215.58s  = Training   runtime
        40.7s    = Validation runtime
Repeating k-fold bagging: 3/20
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 1591.2s of the
1591.19s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4815        = Validation score   (-mean_absolute_error)
        92.05s   = Training   runtime
        43.69s   = Validation runtime
```

```
Fitting model: LightGBM_BAG_L1 … Training model for up to 1553.88s of the
1553.88s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.4787         = Validation score   (-mean_absolute_error)
        101.63s  = Training   runtime
        44.43s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 1513.18s of
the 1513.18s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.1896         = Validation score   (-mean_absolute_error)
        105.4s   = Training   runtime
        1.38s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 1476.17s of
the 1476.17s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8367         = Validation score   (-mean_absolute_error)
        399.43s  = Training   runtime
        1.13s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 1335.99s of the
1335.99s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.9971         = Validation score   (-mean_absolute_error)
        146.79s  = Training   runtime
        29.08s   = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 1278.99s of the
1278.98s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.9213         = Validation score   (-mean_absolute_error)
        63.37s   = Training   runtime
        35.65s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 1251.93s of the
1251.92s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4306         = Validation score   (-mean_absolute_error)
        115.14s  = Training   runtime
        58.36s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 1205.67s of the
1205.67s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.8527         = Validation score   (-mean_absolute_error)
        228.65s  = Training   runtime
```

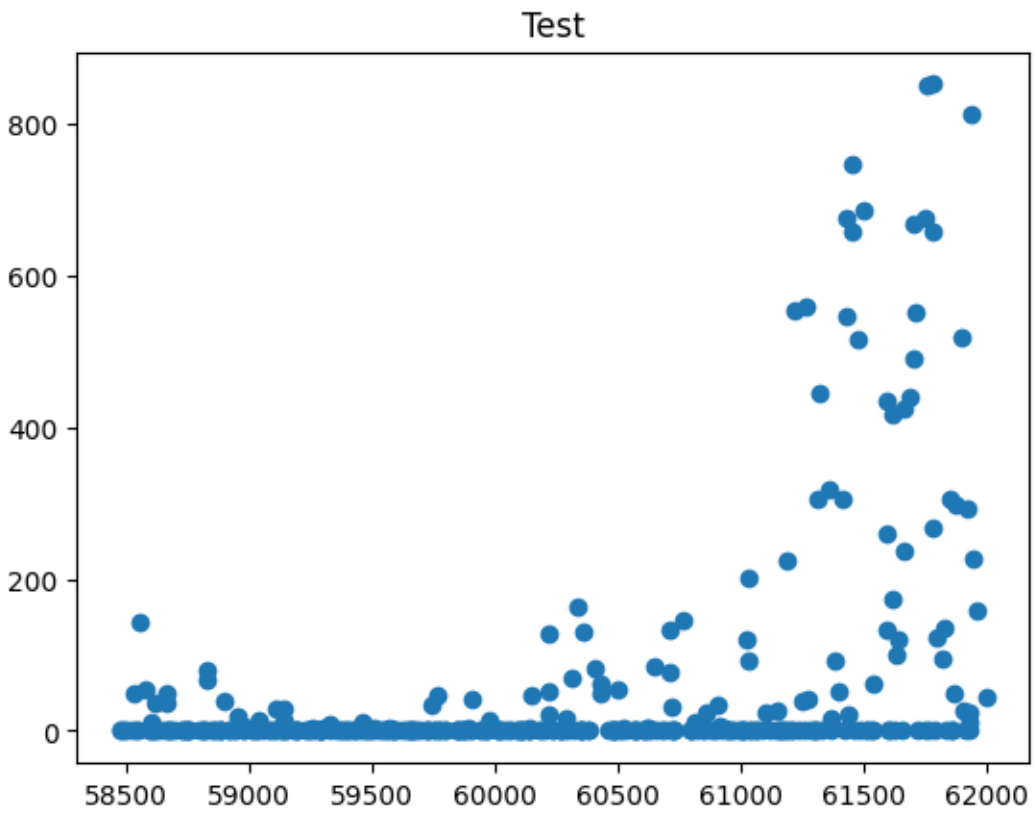
```
        132.81s  = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 1109.05s of the
1109.05s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.482  = Validation score   (-mean_absolute_error)
        160.09s  = Training   runtime
        55.83s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 1043.52s
of the 1043.52s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.1615         = Validation score   (-mean_absolute_error)
        63.32s   = Training   runtime
        0.66s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 1020.9s
of the 1020.9s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.2851         = Validation score   (-mean_absolute_error)
        68.48s   = Training   runtime
        0.69s    = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 996.38s
of the 996.37s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.5784         = Validation score   (-mean_absolute_error)
        96.96s   = Training   runtime
        0.88s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 961.84s
of the 961.83s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.5649         = Validation score   (-mean_absolute_error)
        53.1s    = Training   runtime
        0.62s    = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 942.45s
of the 942.45s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.4038         = Validation score   (-mean_absolute_error)
        235.04s  = Training   runtime
        2.71s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 861.35s
of the 861.34s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.8874         = Validation score   (-mean_absolute_error)
```

```
        240.45s  = Training   runtime
        2.3s     = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 776.07s
of the 776.07s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.2013         = Validation score   (-mean_absolute_error)
        261.75s  = Training   runtime
        2.21s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 685.18s of the
685.17s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.9419         = Validation score   (-mean_absolute_error)
        321.07s  = Training   runtime
        58.51s   = Validation runtime
Completed 3/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 360.0s of the
562.81s of remaining time.
        -12.3391         = Validation score   (-mean_absolute_error)
        0.53s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3037.75s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_122_B/")
Evaluation: mean_absolute_error on test data: -10.723597852022728
        Note: Scores are always higher_is_better. This metric score can be
multiplied by -1 to get the metric value.
Evaluations on test data:
{
    "mean_absolute_error": -10.723597852022728,
    "root_mean_squared_error": -29.434039222380054,
    "mean_squared_error": -866.3626649446074,
    "r2": 0.9627022467439814,
    "pearsonr": 0.9811782810511718,
    "median_absolute_error": -0.5645688772201538
}

Evaluation on test data:
-10.723597852022728
```



	model	score_test	score_val	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L2	-10.723598	-12.339116	4.378928	40.877352	803.763585	0.003843	0.000623	0.532389	2	True	22
1	LightGBM_r118_BAG_L1	-11.007801	-12.921285	2.304665	35.652883	63.369035	2.304665	35.652883	63.369035	1	True	10
2	LightGBMXT_BAG_L1	-11.132298	-13.481468	3.404401	43.693103	92.046542	3.404401	43.693103	92.046542	1	True	3
3	LightGBM_BAG_L1	-11.142879	-14.478723	2.731642	44.433135	101.633064	2.731642	44.433135	101.633064	1	True	4
4	NeuralNetTorch_BAG_L1	-11.182017	-12.836698	0.591085	1.127306	399.426073	0.591085	1.127306	399.426073	1	True	7
5	LightGBMLarge_BAG_L1	-11.261584	-13.941926	9.446423	58.507039	321.065502	9.446423	58.507039	321.065502	1	True	21

```
6          LightGBM_r111_BAG_L1  -11.295378 -13.482024        6.365718
55.826619  160.094988                6.365718               55.826619
160.094988           1       True         13
7           LightGBM_r97_BAG_L1  -11.517776 -13.430644        3.840222
58.361140  115.136348                3.840222               58.361140
115.136348           1       True         11
8        NeuralNetFastAI_BAG_L1  -12.389999 -13.189636        0.506129
1.384786  105.397018                0.506129                1.384786
105.397018           1       True          6
9           LightGBM_r71_BAG_L1  -12.534359 -14.852719        8.406466
132.805016  228.645143                8.406466              132.805016
228.645143           1       True         12
10         LightGBM_r158_BAG_L1  -12.538895 -14.997143        3.252692
29.084249  146.785765                3.252692               29.084249
146.785765           1       True          9
11  NeuralNetFastAI_r145_BAG_L1  -12.685629 -13.403836        0.973206
2.711753  235.039069                0.973206                2.711753
235.039069           1       True         18
12   NeuralNetFastAI_r51_BAG_L1  -12.941687 -13.285118        0.357658
0.688112   68.483178                0.357658                0.688112
68.483178           1       True         15
13         ExtraTreesMSE_BAG_L1  -13.117027 -15.392978        0.447318
0.914828    1.486284                0.447318                0.914828
1.486284           1       True          5
14        ExtraTrees_r19_BAG_L1  -13.312309 -15.257086        0.448604
0.922027    0.965683                0.448604                0.922027
0.965683           1       True          8
15  NeuralNetFastAI_r173_BAG_L1  -14.269517 -13.887434        0.817005
2.300970  240.453202                0.817005                2.300970
240.453202           1       True         19
16   NeuralNetFastAI_r25_BAG_L1  -14.480768 -14.161466        0.359478
0.658887   63.315527                0.359478                0.658887
63.315527           1       True         14
17  NeuralNetFastAI_r128_BAG_L1  -14.515248 -14.201333        0.804592
2.211224  261.745151                0.804592                2.211224
261.745151           1       True         20
18   NeuralNetFastAI_r82_BAG_L1  -14.742232 -14.578388        0.399377
0.882292   96.959816                0.399377                0.882292
96.959816           1       True         16
19  NeuralNetFastAI_r121_BAG_L1  -16.697332 -16.564891        0.297959
0.616057   53.095748                0.297959                0.616057
53.095748           1       True         17
20        KNeighborsDist_BAG_L1  -23.570593 -28.797984        0.016379
0.429177    0.029572                0.016379                0.429177
0.029572           1       True          2
21        KNeighborsUnif_BAG_L1  -24.697224 -28.544405        0.014151
0.389905    0.030971                0.014151                0.389905
0.030971           1       True          1
```



[21]:
```
loc = "C"
predictors[2] = fit_predictor_for_location(loc)
leaderboards[2] = leaderboard_for_location(2, loc)
```

```
Presets specified: ['experimental_zeroshot_hpo_hybrid']
Stack configuration (auto_stack=True): num_stack_levels=0, num_bag_folds=8,
num_bag_sets=20
Beginning AutoGluon training … Time limit = 3600s
AutoGluon will save models to "AutogluonModels/submission_122_C/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Linux
Platform Machine:   x86_64
Platform Version:   #1 SMP Debian 5.10.197-1 (2023-09-29)
Disk Space Avail:   136.15 GB / 315.93 GB (43.1%)
Train Data Rows:    24073

Training model for location C…

Train Data Columns: 44
Tuning Data Rows:    1481
Tuning Data Columns: 44
```

```
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and label-values can't be converted to int).
        Label info (max, min, mean, stddev): (999.6, -0.0, 80.87539, 169.67845)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    130055.27 MB
        Train Data (Original)  Memory Usage: 10.27 MB (0.0% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        41 features in original data used to generate 41 features in processed
data.
        Train Data (Processed) Memory Usage: 8.02 MB (0.0% of available memory)
Data preprocessing and feature engineering runtime = 0.16s …
```

```
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}, {'min_samples_leaf': 1,
'max_leaf_nodes': 15000, 'max_features': 0.5, 'ag_args': {'name_suffix': '_r19',
'priority': 20}}],
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}, {'min_samples_leaf': 5,
'max_leaf_nodes': 50000, 'max_features': 0.5, 'ag_args': {'name_suffix': '_r5',
'priority': 19}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge', {'extra_trees': False, 'feature_fraction': 0.7248284762542815,
'learning_rate': 0.07947286942946127, 'min_data_in_leaf': 50, 'num_leaves': 89,
'ag_args': {'name_suffix': '_r158', 'priority': 18}}, {'extra_trees': True,
'feature_fraction': 0.7832570544199176, 'learning_rate': 0.021720607471727896,
'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args': {'name_suffix': '_r118',
'priority': 17}}, {'extra_trees': True, 'feature_fraction': 0.7113010892989156,
'learning_rate': 0.012535427424259274, 'min_data_in_leaf': 16, 'num_leaves': 48,
'ag_args': {'name_suffix': '_r97', 'priority': 16}}, {'extra_trees': True,
'feature_fraction': 0.45555769907110816, 'learning_rate': 0.009591347321206594,
'min_data_in_leaf': 50, 'num_leaves': 110, 'ag_args': {'name_suffix': '_r71',
'priority': 15}}, {'extra_trees': False, 'feature_fraction':
0.40979710161022476, 'learning_rate': 0.008708890211023034, 'min_data_in_leaf':
3, 'num_leaves': 80, 'ag_args': {'name_suffix': '_r111', 'priority': 14}}],
        'XGB': {},
        'FASTAI': [{}, {'bs': 1024, 'emb_drop': 0.6167722379778131, 'epochs':
44, 'layers': [200, 100, 50], 'lr': 0.053440377855629266, 'ps':
0.48477211305443607, 'ag_args': {'name_suffix': '_r25', 'priority': 13}}, {'bs':
1024, 'emb_drop': 0.6046989241462619, 'epochs': 48, 'layers': [200, 100, 50],
'lr': 0.00775309042164966, 'ps': 0.09244767444160731, 'ag_args': {'name_suffix':
'_r51', 'priority': 12}}, {'bs': 512, 'emb_drop': 0.6557225316526186, 'epochs':
49, 'layers': [200, 100], 'lr': 0.023627682025564638, 'ps': 0.519566584552178,
'ag_args': {'name_suffix': '_r82', 'priority': 11}}, {'bs': 2048, 'emb_drop':
0.4066210919034579, 'epochs': 43, 'layers': [400, 200], 'lr':
```

```
0.0029598312717673434, 'ps': 0.4378695797438974, 'ag_args': {'name_suffix':
'_r121', 'priority': 10}}, {'bs': 128, 'emb_drop': 0.44339037504795686,
'epochs': 31, 'layers': [400, 200, 100], 'lr': 0.008615195908919904, 'ps':
0.19220253419114286, 'ag_args': {'name_suffix': '_r145', 'priority': 9}}, {'bs':
128, 'emb_drop': 0.12106594798980945, 'epochs': 38, 'layers': [200, 100, 50],
'lr': 0.037991970245029975, 'ps': 0.33120008492595093, 'ag_args':
{'name_suffix': '_r173', 'priority': 8}}, {'bs': 128, 'emb_drop':
0.4599138419358, 'epochs': 47, 'layers': [200, 100], 'lr': 0.03888383281136287,
'ps': 0.28193673177122863, 'ag_args': {'name_suffix': '_r128', 'priority': 7}}],
        'CAT': [{}, {'depth': 5, 'l2_leaf_reg': 4.774992314058497,
'learning_rate': 0.038551267822920274, 'ag_args': {'name_suffix': '_r16',
'priority': 6}}, {'depth': 4, 'l2_leaf_reg': 1.9950125740798321,
'learning_rate': 0.028091050379971633, 'ag_args': {'name_suffix': '_r42',
'priority': 5}}, {'depth': 6, 'l2_leaf_reg': 1.8298803017644376,
'learning_rate': 0.017844259810823604, 'ag_args': {'name_suffix': '_r93',
'priority': 4}}, {'depth': 7, 'l2_leaf_reg': 4.81099604606794, 'learning_rate':
0.019085060180573103, 'ag_args': {'name_suffix': '_r44', 'priority': 3}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['RF', 'XGB', 'CAT'] (Specified by `excluded_model_types`)
Fitting 21 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 3599.84s of
the 3599.84s of remaining time.
        -19.8149         = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.33s    = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 3599.28s of
the 3599.28s of remaining time.
        -20.1923         = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.25s    = Validation runtime
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 3598.94s of the
3598.94s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.8238         = Validation score   (-mean_absolute_error)
        29.22s   = Training   runtime
        11.97s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 3565.0s of the
3565.0s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8555         = Validation score   (-mean_absolute_error)
        32.69s   = Training   runtime
        11.1s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 … Training model for up to 3528.27s of the
3528.27s of remaining time.
```

```
        -15.4038        = Validation score   (-mean_absolute_error)
        1.14s    = Training   runtime
        0.77s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 3525.69s of
the 3525.68s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.5826        = Validation score   (-mean_absolute_error)
        31.1s    = Training   runtime
        0.39s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 3493.07s of
the 3493.07s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.6436        = Validation score   (-mean_absolute_error)
        82.85s   = Training   runtime
        0.34s    = Validation runtime
Fitting model: ExtraTrees_r19_BAG_L1 … Training model for up to 3408.61s of
the 3408.61s of remaining time.
        -15.4648        = Validation score   (-mean_absolute_error)
        0.84s    = Training   runtime
        0.79s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 3406.26s of the
3406.25s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.9423        = Validation score   (-mean_absolute_error)
        42.08s   = Training   runtime
        4.03s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 3357.69s of the
3357.69s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.2823        = Validation score   (-mean_absolute_error)
        20.66s   = Training   runtime
        8.37s    = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 3333.23s of the
3333.22s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.7182        = Validation score   (-mean_absolute_error)
        38.59s   = Training   runtime
        13.59s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 3289.77s of the
3289.77s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.3148        = Validation score   (-mean_absolute_error)
```

```
        74.62s   = Training   runtime
        38.76s   = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 3202.85s of the
3202.85s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.9079         = Validation score   (-mean_absolute_error)
        51.5s    = Training   runtime
        19.88s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 3142.8s
of the 3142.8s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.8271         = Validation score   (-mean_absolute_error)
        18.81s   = Training   runtime
        0.18s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 3122.6s
of the 3122.59s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8295         = Validation score   (-mean_absolute_error)
        20.41s   = Training   runtime
        0.2s     = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 3100.76s
of the 3100.76s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.8595         = Validation score   (-mean_absolute_error)
        29.0s    = Training   runtime
        0.24s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 3070.23s
of the 3070.23s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -17.2171         = Validation score   (-mean_absolute_error)
        16.22s   = Training   runtime
        0.2s     = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 3052.51s
of the 3052.51s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.8646         = Validation score   (-mean_absolute_error)
        69.4s    = Training   runtime
        0.76s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 2980.97s
of the 2980.97s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
```

```
        -16.2981        = Validation score   (-mean_absolute_error)
        72.26s   = Training   runtime
        0.73s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 2906.83s
of the 2906.82s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.2728        = Validation score   (-mean_absolute_error)
        79.66s   = Training   runtime
        0.69s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 2825.32s of the
2825.32s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.9072        = Validation score   (-mean_absolute_error)
        103.2s   = Training   runtime
        11.25s   = Validation runtime
Repeating k-fold bagging: 2/20
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 2713.82s of the
2713.82s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.736  = Validation score   (-mean_absolute_error)
        59.15s   = Training   runtime
        23.91s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 2678.25s of the
2678.25s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8129        = Validation score   (-mean_absolute_error)
        63.47s   = Training   runtime
        20.25s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 2641.93s of
the 2641.93s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.5089        = Validation score   (-mean_absolute_error)
        62.11s   = Training   runtime
        0.79s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 2609.14s of
the 2609.14s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.5827        = Validation score   (-mean_absolute_error)
        161.92s  = Training   runtime
        0.67s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 2528.26s of the
2528.26s of remaining time.
```

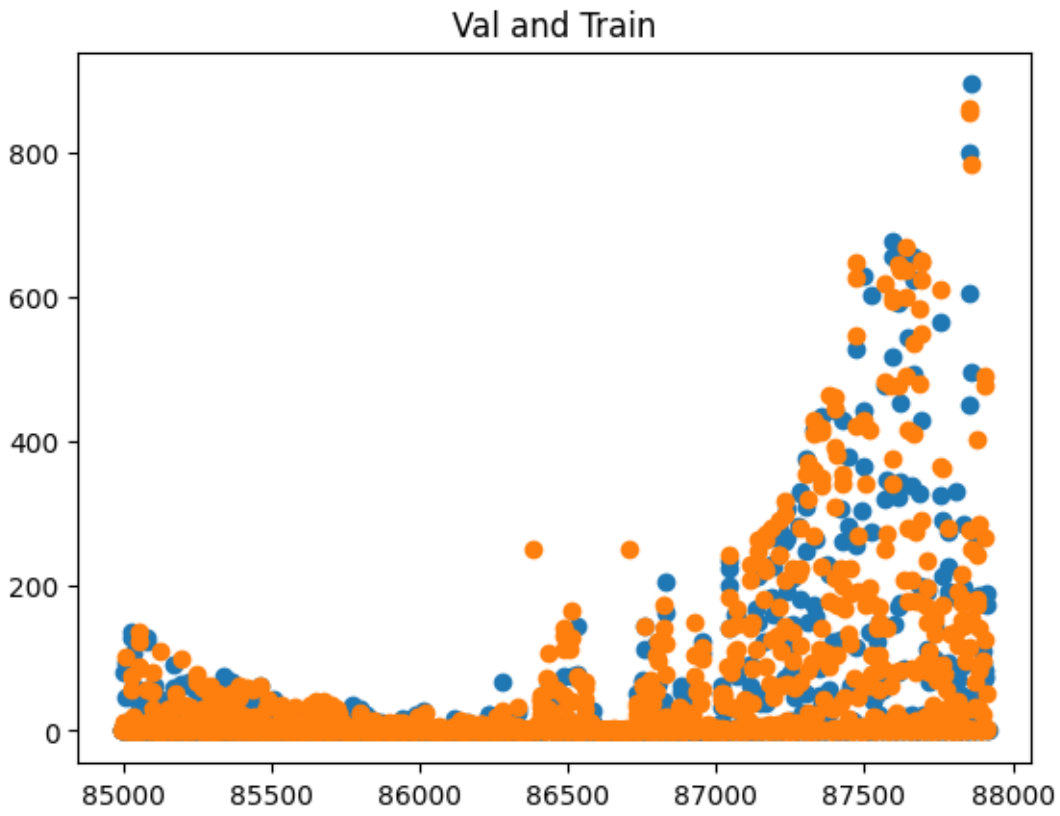
```
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.6379        = Validation score   (-mean_absolute_error)
        96.36s   = Training   runtime
        14.77s   = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 2465.83s of the
2465.83s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.2452        = Validation score   (-mean_absolute_error)
        41.86s   = Training   runtime
        15.46s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 2440.38s of the
2440.38s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.7499        = Validation score   (-mean_absolute_error)
        77.81s   = Training   runtime
        27.73s   = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 2394.07s of the
2394.06s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.2916        = Validation score   (-mean_absolute_error)
        147.84s  = Training   runtime
        80.21s   = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 2303.75s of the
2303.75s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8981        = Validation score   (-mean_absolute_error)
        103.56s  = Training   runtime
        38.56s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 2240.44s
of the 2240.43s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.7878        = Validation score   (-mean_absolute_error)
        37.49s   = Training   runtime
        0.37s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 2220.22s
of the 2220.21s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.6092        = Validation score   (-mean_absolute_error)
        40.56s   = Training   runtime
        0.4s     = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 2198.51s
```

```
of the 2198.51s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.9557        = Validation score   (-mean_absolute_error)
        57.9s    = Training   runtime
        0.5s     = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 2167.71s
of the 2167.7s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -17.2424        = Validation score   (-mean_absolute_error)
        32.46s   = Training   runtime
        0.39s    = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 2149.85s
of the 2149.85s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.5793        = Validation score   (-mean_absolute_error)
        137.72s  = Training   runtime
        1.51s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 2078.85s
of the 2078.84s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.6716        = Validation score   (-mean_absolute_error)
        143.27s  = Training   runtime
        1.42s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 2005.42s
of the 2005.42s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.5392        = Validation score   (-mean_absolute_error)
        154.18s  = Training   runtime
        1.38s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 1928.56s of the
1928.56s of remaining time.
        Fitting 8 child models (S2F1 - S2F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8781        = Validation score   (-mean_absolute_error)
        206.98s  = Training   runtime
        22.89s   = Validation runtime
Repeating k-fold bagging: 3/20
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 1812.72s of the
1812.72s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.7971        = Validation score   (-mean_absolute_error)
        90.66s   = Training   runtime
```

```
        34.91s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 1774.63s of the
1774.63s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.7243         = Validation score   (-mean_absolute_error)
        92.65s   = Training   runtime
        28.48s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 … Training model for up to 1739.2s of
the 1739.2s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.5271         = Validation score   (-mean_absolute_error)
        93.22s   = Training   runtime
        1.21s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 … Training model for up to 1705.97s of
the 1705.96s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.3914         = Validation score   (-mean_absolute_error)
        262.5s   = Training   runtime
        1.03s    = Validation runtime
Fitting model: LightGBM_r158_BAG_L1 … Training model for up to 1603.32s of the
1603.32s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -14.7092         = Validation score   (-mean_absolute_error)
        143.66s  = Training   runtime
        18.97s   = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 … Training model for up to 1546.97s of the
1546.96s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.2132         = Validation score   (-mean_absolute_error)
        62.96s   = Training   runtime
        22.11s   = Validation runtime
Fitting model: LightGBM_r97_BAG_L1 … Training model for up to 1520.5s of the
1520.5s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -11.7531         = Validation score   (-mean_absolute_error)
        115.49s  = Training   runtime
        42.1s    = Validation runtime
Fitting model: LightGBM_r71_BAG_L1 … Training model for up to 1474.45s of the
1474.44s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.3005         = Validation score   (-mean_absolute_error)
```

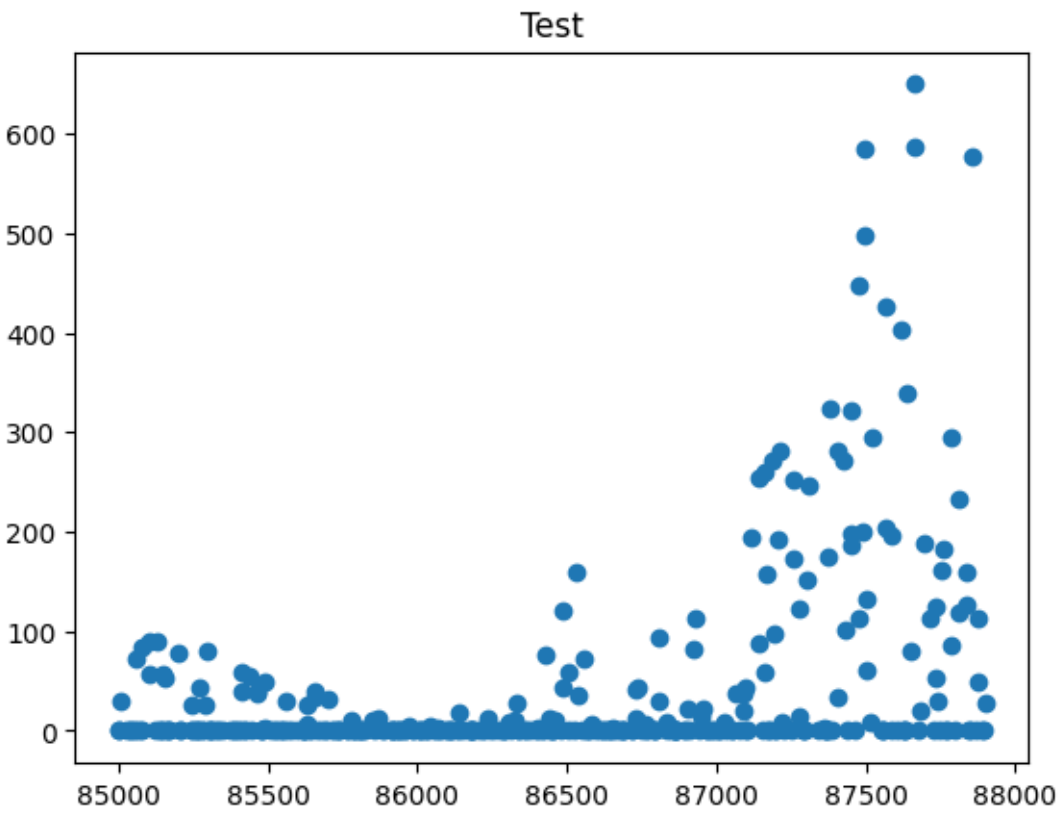
```
        221.62s  = Training   runtime
        119.34s  = Validation runtime
Fitting model: LightGBM_r111_BAG_L1 … Training model for up to 1379.96s of the
1379.96s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.8684         = Validation score   (-mean_absolute_error)
        156.77s  = Training   runtime
        57.32s   = Validation runtime
Fitting model: NeuralNetFastAI_r25_BAG_L1 … Training model for up to 1314.0s
of the 1314.0s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.8597         = Validation score   (-mean_absolute_error)
        56.15s   = Training   runtime
        0.56s    = Validation runtime
Fitting model: NeuralNetFastAI_r51_BAG_L1 … Training model for up to 1293.67s
of the 1293.67s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.5751         = Validation score   (-mean_absolute_error)
        60.52s   = Training   runtime
        0.6s     = Validation runtime
Fitting model: NeuralNetFastAI_r82_BAG_L1 … Training model for up to 1271.97s
of the 1271.97s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -17.0273         = Validation score   (-mean_absolute_error)
        87.02s   = Training   runtime
        0.82s    = Validation runtime
Fitting model: NeuralNetFastAI_r121_BAG_L1 … Training model for up to 1240.79s
of the 1240.78s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -17.2    = Validation score   (-mean_absolute_error)
        48.78s   = Training   runtime
        0.57s    = Validation runtime
Fitting model: NeuralNetFastAI_r145_BAG_L1 … Training model for up to 1222.77s
of the 1222.76s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -13.6868         = Validation score   (-mean_absolute_error)
        206.19s  = Training   runtime
        2.27s    = Validation runtime
Fitting model: NeuralNetFastAI_r173_BAG_L1 … Training model for up to 1151.3s
of the 1151.29s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
```

```
        -16.5738         = Validation score   (-mean_absolute_error)
        210.91s  = Training   runtime
        2.06s    = Validation runtime
Fitting model: NeuralNetFastAI_r128_BAG_L1 … Training model for up to 1080.78s
of the 1080.77s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.8236         = Validation score   (-mean_absolute_error)
        216.36s  = Training   runtime
        1.99s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 … Training model for up to 1015.8s of the
1015.8s of remaining time.
        Fitting 8 child models (S3F1 - S3F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -12.833  = Validation score   (-mean_absolute_error)
        308.54s  = Training   runtime
        35.09s   = Validation runtime
Completed 3/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 360.0s of the
898.84s of remaining time.
        -11.123  = Validation score   (-mean_absolute_error)
        0.53s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 2702.23s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_122_C/")
Evaluation: mean_absolute_error on test data: -11.882573633916262
        Note: Scores are always higher_is_better. This metric score can be
multiplied by -1 to get the metric value.
Evaluations on test data:
{
    "mean_absolute_error": -11.882573633916262,
    "root_mean_squared_error": -28.984463498596043,
    "mean_squared_error": -840.0991243014464,
    "r2": 0.914844370410137,
    "pearsonr": 0.9592036434639014,
    "median_absolute_error": -0.7651807069778442
}

Evaluation on test data:
-11.882573633916262
```




```
                        model  score_test  score_val  pred_time_test
pred_time_val    fit_time  pred_time_test_marginal  pred_time_val_marginal
fit_time_marginal  stack_level  can_infer  fit_order
0           WeightedEnsemble_L2  -11.882574 -11.122967        3.856977
24.068558  386.542620                0.003762                0.000596
0.534351            2       True         22
1          LightGBM_r118_BAG_L1  -11.977019 -11.213198        2.924395
22.111541   62.962651                2.924395               22.111541
62.962651           1       True         10
2            LightGBMXT_BAG_L1  -12.137031 -11.797102        2.805304
34.914344   90.661297                2.805304               34.914344
90.661297           1       True          3
3           LightGBM_r97_BAG_L1  -12.358573 -11.753063        3.886996
42.103179  115.485718                3.886996               42.103179
115.485718           1       True         11
4        NeuralNetFastAI_BAG_L1  -12.998475 -13.527125        0.475597
1.210410   93.215323                0.475597                1.210410
93.215323           1       True          6
5    NeuralNetFastAI_r51_BAG_L1  -13.047249 -12.575052        0.355870
0.598878   60.517148                0.355870                0.598878
60.517148           1       True         15
```

```
6   NeuralNetFastAI_r145_BAG_L1  -13.064247 -13.686793        0.743834
2.268717  206.190490                 0.743834               2.268717
206.190490           1       True         18
7         NeuralNetTorch_BAG_L1  -13.236769 -13.391357        0.555999
1.026419  262.499245                 0.555999               1.026419
262.499245           1       True          7
8               LightGBM_BAG_L1  -13.516948 -12.724342        2.450099
28.478944   92.647044                 2.450099              28.478944
92.647044            1       True          4
9           LightGBM_r71_BAG_L1  -13.549919 -13.300531        8.404623
119.343533  221.619780                 8.404623             119.343533
221.619780           1       True         12
10         LightGBM_r111_BAG_L1  -13.830864 -12.868428        6.638808
57.324214  156.768054                 6.638808              57.324214
156.768054           1       True         13
11         LightGBMLarge_BAG_L1  -14.153453 -12.832983        8.209069
35.091078  308.535153                 8.209069              35.091078
308.535153           1       True         21
12  NeuralNetFastAI_r128_BAG_L1  -15.275111 -16.823553        0.742124
1.988479  216.364365                 0.742124               1.988479
216.364365           1       True         20
13  NeuralNetFastAI_r173_BAG_L1  -15.372847 -16.573827        0.768911
2.060926  210.906921                 0.768911               2.060926
210.906921           1       True         19
14         ExtraTreesMSE_BAG_L1  -15.433922 -15.403829        0.302901
0.769768    1.139719                 0.302901               0.769768
1.139719           1       True          5
15        ExtraTrees_r19_BAG_L1  -15.557787 -15.464786        0.339243
0.786726    0.839571                 0.339243               0.786726
0.839571           1       True          8
16   NeuralNetFastAI_r82_BAG_L1  -15.602025 -17.027349        0.392537
0.819656   87.022309                 0.392537               0.819656
87.022309           1       True         16
17  NeuralNetFastAI_r121_BAG_L1  -15.648391 -17.200041        0.307789
0.565804   48.779081                 0.307789               0.565804
48.779081           1       True         17
18   NeuralNetFastAI_r25_BAG_L1  -15.960404 -16.859700        0.375360
0.561592   56.147095                 0.375360               0.561592
56.147095           1       True         14
19         LightGBM_r158_BAG_L1  -16.031353 -14.709153        2.388238
18.966787  143.661812                 2.388238              18.966787
143.661812           1       True          9
20        KNeighborsUnif_BAG_L1  -20.049167 -19.814903        0.016952
0.331125    0.029224                 0.016952               0.331125
0.029224           1       True          1
21        KNeighborsDist_BAG_L1  -20.130194 -20.192291        0.010500
0.250257    0.027984                 0.010500               0.250257
0.027984           1       True          2
```



[22]:
```
# save leaderboards to csv
pd.concat(leaderboards).to_csv(f"leaderboards/{new_filename}.csv")


for i in range(len(predictors)):
    print(f"Predictor {i}:")
    print(predictors[i].
 ↪info()["model_info"]["WeightedEnsemble_L2"]["children_info"]["S1F1"]["model_weights"])
```

```
Predictor 0:
{'LightGBMXT_BAG_L1': 0.29577464788732394, 'NeuralNetTorch_BAG_L1':
0.352112676056338, 'LightGBM_r118_BAG_L1': 0.16901408450704225,
'NeuralNetFastAI_r51_BAG_L1': 0.028169014084507043, 'LightGBMLarge_BAG_L1':
0.15492957746478872}
Predictor 1:
{'NeuralNetFastAI_BAG_L1': 0.1836734693877551, 'NeuralNetTorch_BAG_L1':
0.3673469387755102, 'LightGBM_r118_BAG_L1': 0.3469387755102041,
'NeuralNetFastAI_r145_BAG_L1': 0.10204081632653061}
Predictor 2:
{'KNeighborsUnif_BAG_L1': 0.03488372093023256, 'NeuralNetTorch_BAG_L1':
0.011627906976744186, 'LightGBM_r118_BAG_L1': 0.8255813953488372,
```

```
'NeuralNetFastAI_r51_BAG_L1': 0.12790697674418605}
```

5 Submit

[23]:
```
import pandas as pd
import matplotlib.pyplot as plt

future_test_data = TabularDataset('X_test_raw.csv')
future_test_data["ds"] = pd.to_datetime(future_test_data["ds"])
#test_data
```

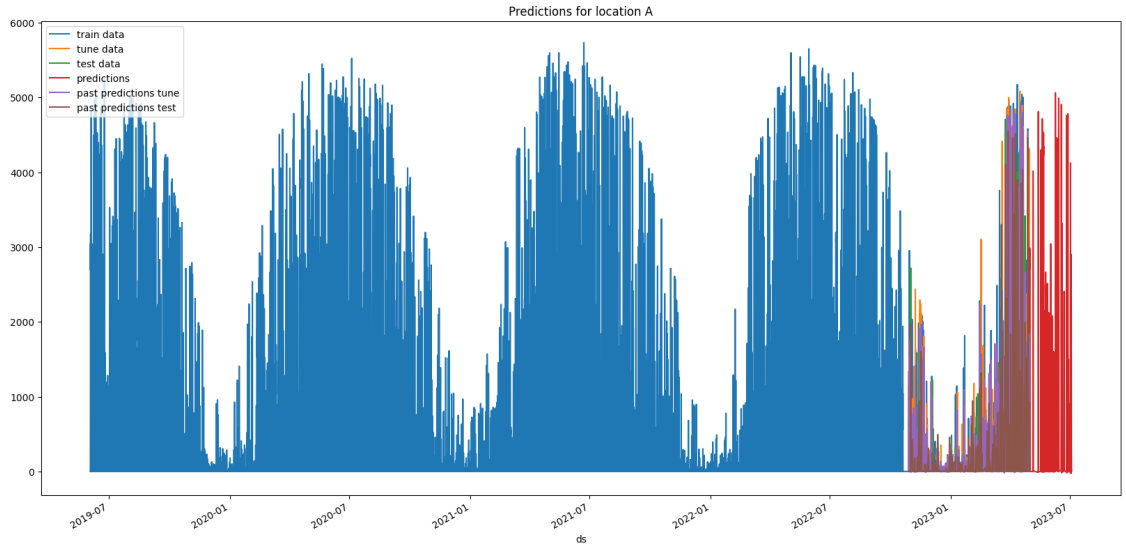
```
Loaded data from: X_test_raw.csv | Columns = 45 / 45 | Rows = 4608 -> 4608
```

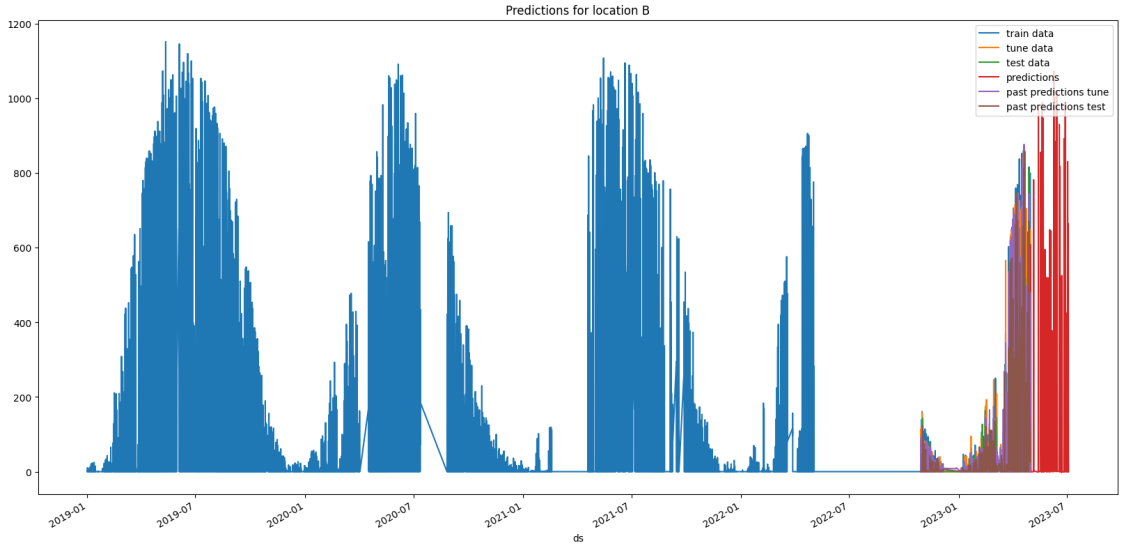
[24]:
```
test_ids = TabularDataset('test.csv')
test_ids["time"] = pd.to_datetime(test_ids["time"])
# merge test_data with test_ids
future_test_data_merged = pd.merge(future_test_data, test_ids, how="inner",␣
 ↪right_on=["time", "location"], left_on=["ds", "location"])

#test_data_merged
```

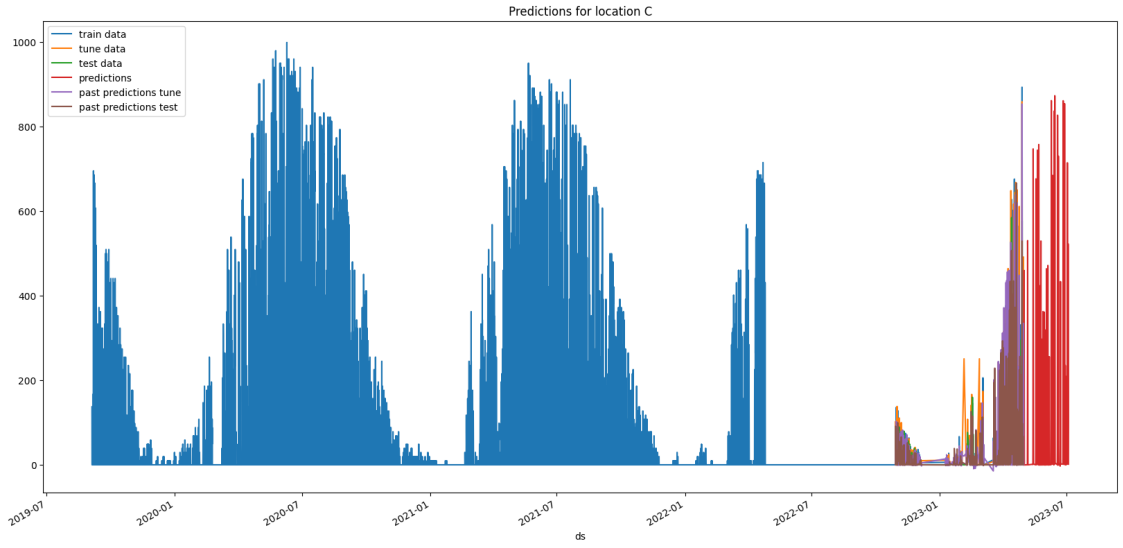
```
Loaded data from: test.csv | Columns = 4 / 4 | Rows = 2160 -> 2160
```

[25]:
```
# predict, grouped by location
predictions = []
location_map = {
    "A": 0,
    "B": 1,
    "C": 2
}
for loc, group in future_test_data.groupby('location'):
    i = location_map[loc]
    subset = future_test_data_merged[future_test_data_merged["location"] ==␣
 ↪loc].reset_index(drop=True)
    #print(subset)
    pred = predictors[i].predict(subset)
    subset["prediction"] = pred
    predictions.append(subset)

    # get past predictions
    #train_data.loc[train_data["location"] == loc, "prediction"] = ␣
 ↪predictors[i].predict(train_data[train_data["location"] == loc])
    if use_tune_data:
        tuning_data.loc[tuning_data["location"] == loc, "prediction"] = ␣
 ↪predictors[i].predict(tuning_data[tuning_data["location"] == loc])
    if use_test_data:
        test_data.loc[test_data["location"] == loc, "prediction"] = ␣
 ↪predictors[i].predict(test_data[test_data["location"] == loc])
```

[26]:
```
# plot predictions for location A, in addition to train data for A
for loc, idx in location_map.items():
    fig, ax = plt.subplots(figsize=(20, 10))
    # plot train data
    train_data[train_data["location"]==loc].plot(x='ds', y='y', ax=ax,␣
 ↪label="train data")
    if use_tune_data:
        tuning_data[tuning_data["location"]==loc].plot(x='ds', y='y', ax=ax,␣
 ↪label="tune data")
    if use_test_data:
        test_data[test_data["location"]==loc].plot(x='ds', y='y', ax=ax,␣
 ↪label="test data")

    # plot predictions
    predictions[idx].plot(x='ds', y='prediction', ax=ax, label="predictions")

    # plot past predictions
    #train_data_with_dates[train_data_with_dates["location"]==loc].plot(x='ds',␣
 ↪y='prediction', ax=ax, label="past predictions")
    #train_data[train_data["location"]==loc].plot(x='ds', y='prediction',␣
 ↪ax=ax, label="past predictions train")
    if use_tune_data:
        tuning_data[tuning_data["location"]==loc].plot(x='ds', y='prediction',␣
 ↪ax=ax, label="past predictions tune")
    if use_test_data:
        test_data[test_data["location"]==loc].plot(x='ds', y='prediction',␣
 ↪ax=ax, label="past predictions test")


    # title
    ax.set_title(f"Predictions for location {loc}")
```







[27]:
```python
temp_predictions = [prediction.copy() for prediction in predictions]
if clip_predictions:
    # clip predictions smaller than 0 to 0
    for pred in temp_predictions:
        # print smallest prediction
        print("Smallest prediction:", pred["prediction"].min())
        pred.loc[pred["prediction"] < 0, "prediction"] = 0
        print("Smallest prediction after clipping:", pred["prediction"].min())
```

```
# Instead of clipping, shift all prediction values up by the largest negative␣
 ↪number.
# This way, the smallest prediction will be 0.
elif shift_predictions:
    for pred in temp_predictions:
        # print smallest prediction
        print("Smallest prediction:", pred["prediction"].min())
        pred["prediction"] = pred["prediction"] - pred["prediction"].min()
        print("Smallest prediction after clipping:", pred["prediction"].min())

elif shift_predictions_by_average_of_negatives_then_clip:
    for pred in temp_predictions:
        # print smallest prediction
        print("Smallest prediction:", pred["prediction"].min())
        mean_negative = pred[pred["prediction"] < 0]["prediction"].mean()
        # if not nan
        if mean_negative == mean_negative:
            pred["prediction"] = pred["prediction"] - mean_negative

        pred.loc[pred["prediction"] < 0, "prediction"] = 0
        print("Smallest prediction after clipping:", pred["prediction"].min())



# concatenate predictions
submissions_df = pd.concat(temp_predictions)
submissions_df = submissions_df[["id", "prediction"]]
submissions_df
```

```
Smallest prediction: -28.481598
Smallest prediction after clipping: 0.0
Smallest prediction: -2.31651
Smallest prediction after clipping: 0.0
Smallest prediction: -3.1237252
Smallest prediction after clipping: 0.0
```

[27]:

	id	prediction
0	0	0.000000
1	1	0.000000
2	2	0.000000
3	3	30.205181
4	4	311.056702
..	…	…
715	2155	62.010193
716	2156	35.529598
717	2157	8.074565

```
718  2158    1.756826
719  2159    1.467647

[2160 rows x 2 columns]
```

[28]:
```
# Save the submission DataFrame to submissions folder, create new name based on␣
 ↪last submission, format is submission_<last_submission_number + 1>.csv

# Save the submission
print(f"Saving submission to submissions/{new_filename}.csv")
submissions_df.to_csv(os.path.join('submissions', f"{new_filename}.csv"),␣
 ↪index=False)
print("jall1a")
```

```
Saving submission to submissions/submission_122.csv
jall1a
```

[ ]:
```
# feature importance
# print starting calculating feature importance for location A with big text␣
 ↪font
print("\033[1m" + "Calculating feature importance for location A..." +␣
 ↪"\033[0m")
predictors[0].feature_importance(feature_stage="original",␣
 ↪data=test_data[test_data["location"] == "A"], time_limit=60*10)
print("\033[1m" + "Calculating feature importance for location B..." +␣
 ↪"\033[0m")
predictors[1].feature_importance(feature_stage="original",␣
 ↪data=test_data[test_data["location"] == "B"], time_limit=60*10)
print("\033[1m" + "Calculating feature importance for location C..." +␣
 ↪"\033[0m")
predictors[2].feature_importance(feature_stage="original",␣
 ↪data=test_data[test_data["location"] == "C"], time_limit=60*10)
```

```
These features in provided data are not utilized by the predictor and will be
ignored: ['ds', 'elevation:m', 'snow_drift:idx', 'location', 'prediction']
Computing feature importance via permutation shuffling for 41 features using 361
rows with 10 shuffle sets… Time limit: 600s…
```

Calculating feature importance for location A…

```
        6689.13s        = Expected runtime (668.91s per shuffle set)
        560.66s = Actual runtime (Completed 8 of 10 shuffle sets) (Early
stopping due to lack of time…)
These features in provided data are not utilized by the predictor and will be
ignored: ['ds', 'elevation:m', 'location', 'prediction']
Computing feature importance via permutation shuffling for 42 features using 361
rows with 10 shuffle sets… Time limit: 600s…
```

Calculating feature importance for location B…

```
        1815.09s        = Expected runtime (181.51s per shuffle set)
        307.41s = Actual runtime (Completed 10 of 10 shuffle sets)
These features in provided data are not utilized by the predictor and will be
ignored: ['ds', 'elevation:m', 'snow_drift:idx', 'location', 'prediction']
Computing feature importance via permutation shuffling for 41 features using 360
rows with 10 shuffle sets… Time limit: 600s…
```

Calculating feature importance for location C…

```
        1336.22s        = Expected runtime (133.62s per shuffle set)
```

[ ]:
```
# save this notebook to submissions folder
import subprocess
import os
#subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
 ↪join('notebook_pdfs', f"{new_filename}_automatic_save.pdf"),␣
 ↪"autogluon_each_location.ipynb"])
subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
 ↪join('notebook_pdfs', f"{new_filename}.pdf"), "autogluon_each_location.
 ↪ipynb"])
```

[ ]:
```
# import subprocess

# def execute_git_command(directory, command):
#     """Execute a Git command in the specified directory."""
#     try:
#         result = subprocess.check_output(['git', '-C', directory] + command,␣
 ↪stderr=subprocess.STDOUT)
#         return result.decode('utf-8').strip(), True
#     except subprocess.CalledProcessError as e:
#         print(f"Git command failed with message: {e.output.decode('utf-8').
 ↪strip()}")
#         return e.output.decode('utf-8').strip(), False

# git_repo_path = "."

# execute_git_command(git_repo_path, ['config', 'user.email',␣
 ↪'henrikskog01@gmail.com'])
# execute_git_command(git_repo_path, ['config', 'user.name', hello if hello is␣
 ↪not None else 'Henrik eller Jørgen'])

# branch_name = new_filename

# # add datetime to branch name
# branch_name += f"_{pd.Timestamp.now().strftime('%Y-%m-%d_%H-%M-%S')}"

# commit_msg = "run result"
```

```
# execute_git_command(git_repo_path, ['checkout', '-b',branch_name])

# # Navigate to your repo and commit changes
# execute_git_command(git_repo_path, ['add', '.'])
# execute_git_command(git_repo_path, ['commit', '-m',commit_msg])

# # Push to remote
# output, success = execute_git_command(git_repo_path, ['push',␣
 ↪'origin',branch_name])

# # If the push fails, try setting an upstream branch and push again
# if not success and 'upstream' in output:
#     print("Attempting to set upstream and push again...")
#     execute_git_command(git_repo_path, ['push', '--set-upstream',␣
 ↪'origin',branch_name])
#     execute_git_command(git_repo_path, ['push', 'origin', 'henrik_branch'])

# execute_git_command(git_repo_path, ['checkout', 'main'])
```

[ ]: